ISSN 0998-9250 Février 1991 / N° 10

Avec cahier Dr Dobb's Journal

DEBUG magazine

FETE SES 2 ANS

NOUVEAU
ABONNEMENT 2 ANS
50 % DE REMISE

NOUVEAU
ABONNEMENT DISQUETTES
LISTINGS

NOUVEAU
MARCHE PC
PETITES ANNONCES PRO

NOUVEAU
SERVEUR TELEMATIQUE
3615 DEBUG

M 3032 - 10 - 37,00 F

Belgique : 258 FB - Suisse : 11,5 FS

COMPUMART

21, rue de Fécamp - 75012 Paris

Les prix discount !

LANGUE FORMATP RODUIT	Prix HT	TTC
ABOVE SOFTWARE		
US + ABOVE DISK	450.34	534
ALDRIDGE COMPANY		
US 5 PRECURSOR	872.24	1034
US 5 TREE 86	464.56	551
BORLAND		
FR-REFLEX+WORKSHOP	722.01	856
FR - TURBO BASIC	722.01	856
FR 5 WORKSHOP (REFLEX)	401.12	476
BOURBAKI		
US 5 MUSCLE	474.04	562
US 5 SFE UTILITIES	398.20	472
BRIDGEWAY PUBLISHING		
US 5 FAST TRAX	369.76	439
US 5 MENU BLOCK	328.19	389
BRODERBUND		
US 3 CALCULUS MAC	284.43	337
US 5 VCR COMPANIONAPPLE2	328.19	389
BUMBLE SOFTWARE		
US 5 DB FAST COMPILER	587.82	697
CENTRAL POINT SOFTWARE		
FU 5 PCTOOLS DELUXE F	284.43	337
US 3 PCTOOLS DELUXE MAC	284.43	337
US + PCTOOLS DELUXE	538.15	638
US + PCTOOLS DELUXE	703.77	835
COMPUSERVE		
US COMPUSERVE STARTER	189.62	225
DAC EASY		
US 5 LUCID 3 D	474.04	562
DATA ACCESS		
US 5 DATAFLEX MULTIPC	5925.56	7028
US 5 DATAFLEX RUNTIME	1564.35	1855
DATA TRANSFORMS		
US 5 FONTRIX	1185.11	1406
US 5 PRINTRIX	1185.11	1406
DATASTORM TECHNOLOGIES		
US 5 PROCOMM +	474.05	562
DESIGN SOFTWARE		
US 5 DS BACKUP +	189.62	225
US 5 DS BACKUP ANSI	189.62	225
US 3 DS BACKUP MAC	189.62	225
US 5 DS OPTIMIZE	189.62	225
US 5 DS RECOVER	189.62	225
US 5 DS TUTOR	189.62	225
US 5 WORD FOR WORD	237.02	281
DIA COM		
UD 5 DATA BRIDGE	510.51	605
DYSAN		
5DISQUE ALIGNEMENT AT	274.95	326
5DISQUE ALIGNEMENT PC	274.95	326
EXECUTIVE SYSTEMS		
US + XTREE PRO NET	1827.48	2167
FICHER INNS		
US 5 WATCH DOG	1791.89	2125
FOX SOFTWARE		
US 5 FOX BASE+ MONO	1763.45	2091
US 5 FOX BASE+ MULTI	2654.65	3148
FUNK SOFTWARE		
FR 5 SIDEWAYS	692.84	822
GAZELLE		
US 5 OPTUNE	464.56	551
GOLDEN BOW		
US 5 V OPTIMIZE	426.64	506
HILGREAWE		
US 5 ACCESS	341.31	405
IMSI		
FR 5 DPG	1057.48	1254
IS2		
FR + TURBOTEXTE	692.84	822
ISOGON		
US 5 NEW SPACE	211.50	251
J&M SYSTEMS		
US 5 J&M MACH3 (SOFT)	142.21	169
US LECTEUR 3"1/2 EXT	1516.94	1799
US + MEMORY MINDER	331.83	394
LASERGO		
US 5 GOSCRIPT	1374.73	1630
LOTUS		
FR - LOTUS 1-2-3	2552.55	3027
FR - SYMPHONY	1823.25	2162
MERIDIAN TECHNOLOGY		
US 5 DEJAVIEW	521.45	618
MGI		
US 5PUBLISHER PICTUREPAK	616.26	731
MICRO SOLUTIONS		
US 5 MATCH MAKER MAC	1232.52	1462
US 5 MATCH POINT APPLE	1166.15	1383
US 5 UNIFORM PC	540.41	641
MICROPRO		
FR 5 SUPER SORT	619.91	735
MICROSOFT		
US 5 MACRO ASSEMBLER	815.36	967
FR 5 RBASE	2114.97	2508
MICROTEX / MACE		
US 5 GRASP	1061.86	1259
MOSAIC		
US 5 TWIN CLASSIC	692.84	822
MULTISOFT		
US - PC KWIK POWERPAK	694.51	824
MYCROFT LABS		
US 5 MAXI MYTE	328.19	389
NEW ENGLAND SOFTWARE		
US + GRAPH IN THE BOX	625.74	742
NORTON		
US + NORTON ADVANCED	546.97	649
US 5 NORTON COMMANDER	284.43	337
US + NORTON COMMANDER	503.22	597
US + NORTON GUIDE ASSEMBL	485.93	576
US + NORTON GUIDE BASIC	485.93	576
US + NORTON GUIDE C	485.93	576
US + NORTON GUIDE OS/2	727.70	863
US + NORTON GUIDEPASCAL	485.93	576
US 5 NORTON UTILITIES ADV	328.19	389
US 5 NORTON UTILITIES STD	255.25	303
US + NORTON UTILITIES STD	474.04	562
NOVELL		
US 5 NETWARE ELS 1	3930.93	4662
ONTRACK		
US 5 DISK MANAGER	564.11	669
US 5 DOSUTILS	503.22	597
POWER LINE		
US 5 SOURCE PRINT	663.66	787
US 5 TREE DIAGRAMMER	483.53	573
QUAID		
US - ANTIDOTE	284.43	337
QUARTERDECK		
US 5 DESQVIEW	668.40	793
US 5 DESQVIEW COMPANION	559.37	663
US 5 QEMM 50/60	312.87	371
SDI		
US 5 RETREEVE	502.49	596
SIMON & SCHUSTER		
US 5 WEBSTER'S NEW WORLD	654.18	776
SOFTCRAFT		
US 5 FANCY FONT	1061.86	1259
US 5 FONTWARE INSTALL.	255.25	303
SOFTLOGIC		
US 5 CUBIT	398.20	472
US 5 DOUBLE DOS	353.20	419
US 5 MAGIC MIRROR	559.37	663
US 5 SOFTWARE CAROUSEL	402.94	478
SOFTSTYLE		
US 5 PRINTWORKS LASER	711.07	843
SPINNAKER		
US + PIZAZZ PLUS	748.99	888
STORAGE DIMENSIONS		
US 5 SPEEDSTOR UNIVERSAL	331.83	394
UNISON WORLD		
US 5 ART GALLERY 1 & 2	213.32	253
US 5 FONT & BORDERS	208.58	247
US + PRINT MASTER +	284.43	337
US 5 ART GALLERY I	160.45	190
WORDTECH SYSTEMS		
US 5 NETWORKER PLUS	1301.29	1543
Z SOFT		
US 5 FONT PAK B1 & S1	502.49	596
DISQUETTES VERBATIM(par 10)		
3 DATALIFE 3"1/2 2SHD	80.22	95
5 DATALIFE 5"1/4 2SDD	43.76	52
5 DATALIFE 5"1/4 2SHD	58.34	69

CONDITIONS DE VENTE :

1. Prix nets, frais de livraison inclus
2. Livraison dans la limite de stocks en 48 heures
3. Paiement seulement par chèque, carte de crédit (Visa, CB. MC) ou mandat
4. A cause de lois sur le copyright, pas de reprise de logiciels ouverts

Je veux recevoir la liste complète des prix.

Nom _____

Société _____

Adresse _____

Ville _____

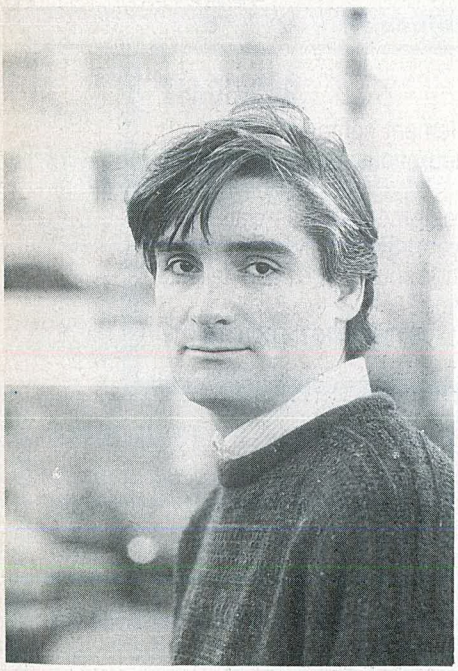
Code Postal |_|_|_|_|_| Tél. : _____

AUJOURD'HUI C'EST DEJA DEMAIN

EDITO

DEBUG MAGAZINE fête ses deux ans. Lancé lors du PC FORUM 1989, le premier magazine technique français a réussi a survivre - pour vous et grâce à vous. Car, comme vous pouvez le remarquer, DEBUG MAGAZINE n'a presque aucune recette publicitaire. Le prix que vous payez en kiosque (diminué de plus de la moitié de frais retenus par le distributeur) nous suffit pour produire numéro après numéro et équilibrer les comptes.

Je salue ici l'effort de l'équipe actuelle et ceux qui ont contribué dans le passé. Actuellement, nous sommes quatre. D'abord l'équipe administrative : une comptable (abonnements, disquettes), un vendeur d' espace publicitaire. Ensuite, la rédaction : la maquettiste et votre serviteur. Trois personnes contribuent de manière permanente de l'extérieur.Les autres collaborateurs - c'est vous, tous ceux qui écrivent des programmes ou commentaires et nous les envoient pour la publication. En effet, l'espace rédactionnel deDEBUG MAGAZINE est le vôtre : des utilisateurs avertis, des techniciens, des programmeurs. Aujourd'hui, vous êtes quelques 1.500 abonnés et 12.000 lecteurs qui nous achètent régulièrement en kiosque, en France, Belgique, Antilles et en Afrique.



L'Avenir ? Nous voulons vous fournir un magazine indépendant, de meilleure qualité, à la pointe de la technologie du jour, à la frontière entre la découverte du présent et la créativité qui fonde l'avenir. C'est ce fameux « espace de curiosité informatique » - et le droit d'auteur, le droit de l'utilisateur vous en faîtes partie. Nous voulons aussi défendre la « Res Publica », la liberté de l'individu à créer, la propriété intellectuelle et le droit d'auteur, le droit de l'utilisateur. Pour qu'un vendeur peu scrupuleux ne puisse pas marchander vos partagiciels. Pour que le compilateur que vous achetez soit livré sans bugs, ni plombage qui nuisent à son utilisation.

Et nous voulons toujours mieux vous servir. Des abonnements groupés à un prix encore plus réduit rendent DEBUG MAGAZINE encore moins cher. Un serveur télématique 3615 DEBUG pour télécharger rapidement des listings, laisser des petites annonces, échanger des idées - nous attendons une ligne TRANSPAC pour le démarrer. Et un nouvel espace de la communication : MARCHE PC. En effet, rares sont les développeurs, consultants, ateliers de maintenance ou sociétés de service qui puissent payer une page de publicité en couleur. Le prix des modules dans le MARCHE PC rend enfin la communication abordable pour tous. N'oubliez pas - vous pouvez toujours échanger l'achat de l'espace publicitaire contre une traduction ou un article technique de qualité.

Vos téléphones et lettres, de critiques et de compliments, nous encouragent pour continuer, je vous en remercie. Je vais continuer de tout faire pour mériter votre confiance

Artur G Piotrowski

Disk Explorer — 895,00 HT

L'éditeur de tous vos supports magnétiques, capable, par la simple frappe d'une touche, de récupérer un fichier accidentellement effacé.
- Modifiez le contenu des disques et disquettes
- Affichez les répertoires et changez les attributs des fichiers même cachés.
- Copiez les données d'une zone du disque vers une autre ou sur un disque différent.
- Recherchez une chaîne de caractères sur un support.
- Analysez les disquettes au format non-DOS (chaque octet d'une piste, les octets de synchronisation, les marques d'adresses des ID et des données, les zones de données, CRC, etc...)
- Créez votre propre format.

Analyzer II — 1695,00 HT

Le super «debugger» qui vous montre en clair l'exécution de votre programme compilé.
- Visualisez le contenu de la mémoire sous forme d'instructions et déplacez-vous dans l'écran aussi facilement qu'avec un éditeur de textes.
- Constatez les changements à chaque interruption.
- Suivez l'exécution de votre programme même protégé contre l'analyse et même si vous n'en avez plus les sources.
- Visualisez les étapes d'initialisation (boot)
- Analysez toutes les entrées/sorties de votre PC ainsi que l'échange d'informations entre le programme et les ports série ou parallèle.

CopyWrite — 895,00 HT

Faites les copies de sauvegarde de vos programmes originaux.

Antidote — 745,00 HT

Protégez votre PC contre les virus avant qu'ils n'agissent

FAITES DES ÉCONOMIES !!

Quaid Pack Professionnel (Analyzer + Disk Explorer) — **2072,00 HT**

Quaid Pack Expert (Analyzer + Disk Explorer + CopyWrite) — **2788,00 HT**

Mise à jour CopyWrite — **395,00 HT**

Ces produits sont livrés dans leur présentation originelle accompagnée d'une traduction du manuel

DR DOS 5.0 (Français) — 990,00 HT

Voilà enfin le système d'exploitation monotâche complet et parfaitement compatible avec vos applications et structures de disques, de xx-DOS à xx-DOS 4.01. Mieux, vous pourrez à nouveau utiliser des applications qui ne marchent plus dans xx-DOS 4.nn.
Pour moins de 1000 Francs HT, DR DOS 5.0 est en Français proprement écrit et clairement documenté. Il supporte les réseaux locaux (MS-NET, Novell Netware, 3COM, IMB PC Lan et Banyan Vines), possède une interface graphique conviviale supportant la souris, un éditeur de texte en mode pleine page, un outil de transfert de fichiers (115000 bauds).
DR DOS est le seul système d'exploitation DOS offrant jusqu'à 620 kO de mémoire disponible (PC à base de 80286, i386 et i486)

Option Board Deluxe v. 5.4 — 1395,00 HT

L'outil de sauvegarde le plus puissant !
- Exploitez les disquettes MAC (données) sur votre PC avec son lecteur 3,5" interne.
- Faites vos copies de sauvegarde de vos originaux, même si les disquettes ne sont pas au format DOS (APPLE, ATARI, KEYPRO ou AMIGA)
- Grâce à l'éditeur binaire, analysez et modifiez les enregistrements magnétiques sur disquettes (zones de données et de synchronisation incluses)

Si vous possédez PC-TOOLS, vos sauvegardes avec PC-BACKUP se feront encore plus rapidement.
Option Board est livré accompagné d'une traduction de son manuel.

PC TOOLS Deluxe v. 6 Fra. — 1350,00 HT

Cette nouvelle version regroupe tous les utilitaires DOS les plus puissants ainsi que les outils de sauvegarde, un gestionnaire de bureau et des programmes d'optimisation des performances.

- Vous pourrez visualiser vos fichiers en format natif (plus de 30 modes)

- Localisez des fichiers par nom ou contenu : l'exécution du programme associé se fera automatiquement.

- Accès direct aux commande du DOS (ligne de commande)

- Et encore de nouveaux outils !!

LOGIQUE Direction Commerciale : 14, Rue de Plaisance - 75014 PARIS Tél. : (1) 45.39.69.18 - Fax (1) 45.41.47.96
Siège Social : 36, cours du Chapeau Rouge - 33000 BORDEAUX Tél. 56.44.21.11 - Fax : 56.51.73.94

Les Prix en rouge sont promotionnels jusqu'au 31/12/90	Qté	Prix Unitaire H.T.	Montant H.T.
DR DOS 5.0 (Français)		990,00 HT	
Disk Explorer		895,00 HT	
Analyzer II		1695,00 HT	
CopyWrite		895,00 HT	
Mise à jour CopyWrite		395,00 HT	
Antidote		745,00 HT	
Quaid Pack Professionnel (Analyzer + Disk Explorer)		2072,00 HT	
Quaid Pack Expert (Analyzer + Disk Explorer + CopyWrite)		2788,00 HT	
PC TOOLS Deluxe v. 6 Français		1350,00 HT	
Option Board Deluxe v. 5.4		1395,00 HT	

Nom _____ Prénom _____
Adresse _____
C.P. _____ Ville _____ Tél. _____
Je Joins à ma commande mon règlement ☐ par chèque ☐ Par Carte Bleue N° _____
Date d'expiration _____ DATE _____ SIGNATURE

Total Produits	
Forfait Port	+ 40,00
Total H.T.	
T.V.A. 18,60%	
Total T.T.C.	

[SERVICE LECTEURS : INTERESSANT = 1 / MOYEN = 2/ MAUVAIS = 3]

DEBUG MAGAZINE

Rédacteur en Chef
Directeur de Publication
Artur G. Piotrowski

Maquettiste
Elisa Le Moal

Ont collaboré à ce numéro
ART, Serge Grun, Bruno Vasseur,
Pierre Morel Fourrier,
Georges Bourne,
Randall L. Hyde, ,
E.M. Amblard,

Iconographie
Photos : Image Bank, Debug Magazine

REDACTION
TELEMATIQUE
VENTE AU NUMERO
SERVICE LECTURE
GESTION DES ABONNEMENTS
EDITEUR
SOFTMEDIA
SARL de presse au c[al] de 5000 F.
3, rue de l'Atlas
75019 Paris
Téléphone : (1) 42 02 29 16
Télécopie : (1) 42 45 58 13

REGIE PUBLICITAIRE
Dlirecteur de Publicité
Alexandre Jacob
Agen*s Commerciaux
Andrée Moritz-Cickowlas

COMPTABILITE - ABONNEMENTS
Lidia Zakan

Prix de l'abonnement
11 numéros + 1 numéro spécial
486 F. T.T.C. (France)
Prix au numéro : 37 F.

IMPRIMERIE, PHOTOGRAVURE
GC Conseil

Mise en page : X.PRESS-MAC

Dépôt légal à la parution
Commission paritaire 71290
ISSN 0998-9250
membre inscrit à O.J.D.

« La loi du 11 Mars 1957 n'autorisant pas aux termes des alinéas 2 et 3 de l'article 41 d'une part que les copies ou reproductions strictement réservées à l'usage privée du copiste et non destinés à une utilisation collective et, d'autre part, que les analyses et les courtes citations dans un but d'exemple et d'illustration, faites sans le consentement de son auteur ou de ses ayant droit ou ayant cause, est illicite (alinéa premier de l'article 40). Cette représentation ou reproduction, par quelque procédé que ce soit, constituerait donc une contrefaçon sanctionnée par les articles 425 et suivants du code pénal.

SOMMAIRE

N° 10 - FEVRIER 1991

EDITO
Aujourd'hui, c'est déjà demain **3**

RUMEURS **8**

DOSSIER Dr. DOBB'S
La Programmation orientée Objet
en Assembleur **10**
Kermit pour O/S 2 **18**

PROGRAMMATION
Cahier Listing **23**

DOSSIER
Procédures de Sortie en Turbo Pascal **57**
Turbo C++ et Zortech ++ **60**
Protect **65**

PODIUM : Des Outils indispensables **69**
IDEE : Sentinelle Automatique **72**
ANNONCES **74**

PARLONS - EN

Joignez le dialogue!

Ecrivez nous à l' adresse suivante :

DEBUG MAGAZINE
3 rue de l'Atlas
75019 PARIS

ou à notre **ID** sur **COMPUSERVE :** 71021,1666

Toutes les lettres deviennent la propriété de DEBUG MAGAZINE et ne peuvent pas être retournées. Nous nous réservons le droit d'abréger le texte pour des raisons d'espace (c'est-signalé par (...)).
Artur G Piotrowski

CHAOS

☞ ***« Il me semble que vous « meublez »" votre revue avec des articles très éloignés de l'informatique; par exemple celui sur Hubble et sur Chaos. Bien qu'ayant lu et apprécié ces articles, ce n'est pas pour ça que j'achète DEBUG.***

- Dans le n° 1 vous parliez de DEBUG, qui est à l'origine du nom de votre revue, et depuis, rien.

- Dans le dernier numéro vous commencez à parler du MAC; je pense que DEBUG devrait être concentré uniquement sur PC, PS2...»
Philippe Antras, Marseille

☞ ***« BRAVO pour le petit dossier « CHAOS » (20/20), un prodige de clarté. Même mon épouse (prof de lettres, justement), a pu le lire! J'attends un petit miracle sur les fractales ».***
Jean Claude Bertucelli, Monteux

☞ ***« Je vous confirme par la présente lettre mon intérêt, et l'intérêt vraisemblable de nombreux collègues, pour un logiciel permettant de simuler sur ordinateur des phénomènes relevant du « Chaos déterministe », et en particulier permettant de tracer différents attracteurs étranges. Le programme « Chaos sur votre ordinateur » publié dans le numéro de janvier me parait en ce sens particulièrement intéressant. Malheureusement, je ne suis pas en mesure d'installer effectivement ce programme sur mon ordinateur (..), en effet, les connaissances en programmation me manquent pour employer utilement le listing qui a été publié, et dont vous m'avez dit qu'il constituait un simple squelette pour ces applications ».***
Dr B Gibello, Paris

Selon un concept rédactionnel bien prémédité, dans chaque numéro de DEBUG MAGAZINE nous nous efforçons d'aborder un champ le plus large de ce que j'appelle « la curiosité informatique ». DEBUG n'est pas un simple « cahier à listing PC », bien que cela serait la formule la plus simple. Nous ne voulons pas vous limiter à une quelconque plate-forme matérielle ou logicielle, langage ni taille de la machine. DEBUG veut rester ouvert à tout phénomène lié à la programmation, ses origines, le processus de la création même, ses résultats et interactions avec le monde « extérieur ». En somme, nous voulons satisfaire votre besoin de curiosité intellectuelle, qui s'exprime et se matérialise par vos agissements informatiques.

C'est cette soif de savoir et de faire, qui a été à l'origine de l'informatique personnelle et qui est à la source de la découverte, du progrès technologique, des toutes les bouleversements que le monde actuel subit - parfois bons, parfois néfastes. Preuves? Pour ne parler que du MAC, c'est son environnement graphique qui a donné l'envie à plusieurs auteurs de le reproduire sur PC: VISION, GEM, WINDOWS, PM, DESQview/X. Les deux éditoriaux que nos avons consacré à la guerre dans le Golfe, avant qu'elle n'éclate, se vérifient aujourd'hui dans la réalité. C'est grâce à une immense supériorité informatique des forces alliées que nos pertes sont si limitées et nous pouvons espérer une fin rapide et favorable. Sans les logiciels de traitement d'image, le défaut du miroir d'Hubble serait fatal - actuellement le télescope spatial, bien que « myope », fournit une quantité d'informations insoupçonnables, qui révolutionnent notre savoir sur l'Univers. L'informatique est partout, mais ce n'est pas elle qui est la plus importante, elle n'est qu'un moyen d'expression pour les humains.

Les programmes ne sont qu'un reflet de la personnalité, des besoins ou de la fantaisie des personnes. A part des solutions concrètes, des exemples pratiques, nous vous rapportons les faits qui font évoluer l'industrie des logiciels, nous découvrons avec vous les gens et les motifs qui les font agir, créer tel ou tel programme.

Evidemment, puisque les textes sont écrits, en majorité, par vous, DEBUG reste prés du standard PC, que vous utilisez le plus souvent au travail ou à la maison. Revenant au sujet « CHAOS », et compte rendu de votre demande, nous allons publier des programmes concrets de divers attracteurs étranges. Et si quelqu'un veut partager avec les lecteurs ses expériences avec DEBUG (ou SID), nos colonnes restent ouvertes.

Un journal ne peut pas être un manuel, un paradigme, ni un « tube » à enfoncer des idées acquises. DEBUG se veut être un champ d'échange des idées, une part de cette « CYBERSPACE » avancée, don parlent notamment les pères de l'informatique personnelle, Mitch Kapor et John Perry Barlow, fondateurs de l'association « Electronic Frontier Foundation ». Ce dernier dit : « la plupart de gens à qui je parle de ces choses me disent « je suis l'architecte des connecteurs de cartes, et non un philosophe social ». Et je le comprends fort bien. Mais la société est tellement pénétrée par des donnes technologiques incertaines que même un constructeur de fonds de paniers est mieux qualifié pour parler d'elle que plupart des gens. C'est notre rôle de parler de l'esthétique et de la culture.

plupart des gens. C'est notre rôle de parler de l'esthétique et de la culture. Jaron Lanier a dit qu'il n'y a rien de technologiquement avancé dans le MAC: « c'est un appareil culturel ». Il a raison. DEBUG MAGAZINE vous appartient, il est fait par vous et pour vous. Alors, à vos plumes, ou plutôt, à vos claviers!

☞ ***« Ayant réussi avec succès la représentation graphique de l'attracteur de E. Lorenz et de M. Henon, il m'est impossible de tracer celui de Gumowski-Mira. Pourriez-vous m'indiquer de nouveau la formule ainsi que les valeurs initiales de x,y,a,m ».***
Frederic Urbain, Lille

La formule et les valeurs de m et a sont corrects. Essayez avec x=9 et y=0; ils doivent avoir des valeurs différentes pour que l'attracteur « accroche ». Il nous est difficile de vous conseiller sans le listing de vos programmes, d'ailleurs, nous serions heureux de les recevoir pour les communiquer à d'autres lecteurs qui ont eu des difficultés dans la transcription du pseudo-code en langage concret.

COQUILLES

☞ ***« Votre réponse à Mr Pascal Menoud m'a fait grincer des dents (je suis prof !). L'Education Nationale a bon dos, que faire avec des élèves qui regardent, en moyenne, la TV 25 à 30 heures par semaine ? Qui téléphonent ou enregistrent au lieu d'écrire ? Si vous avez des idées, je suis preneur ».***
Jean Claude Bertucelli, Monteux

☞ ***« J'ai été effaré par le nombre de coquilles de toutes sortes qui émaillent le texte. (...) A la page 14, par exemple, en une demi-page, j'ai relevé huit fautes, allant de la duplication de phrase à la faute de français ou d'orthographe. Dans le tableau de la page 16, j'ai pu rectifier quelques erreurs (disposition de la formule de la ligne 1, ligne 2 manque la barre horizontale du signe racine carrée, ligne 4 x=-1 à la place de x=1) (...) Il n'est d'ailleurs pas interdit de retourner un article à un auteur en le priant de le présenter convenablement. (...) Mon expérience de correcteur me fait également être en désaccord avec la boutade dans la même réponse selon laquelle toute relecture est une source inépuisable d'erreurs. Et qu'une relecture demande un délai de trois semaines au moins ».***
Dr B Gibello, Paris

Le problème de la perfection littéraire de DEBUG tient coeur à tout le monde vous, les lecteurs, qui nous signalez les défauts et nous, ici, qui sommes responsables du résultat. Depuis le numéro précédent (DM 9) nous avons mis en oeuvre un certain nombre de mesures qui doivent progressivement améliorer l'état des choses. Tout d'abord, les traductions sont actuellement effectuées par des personnes qui sont compétentes tant du point de vue technique que du point de vue littéraire. Vous devez constater une nette amélioration. Nous cherchons les correcteurs compétents pour les textes crées en France et nous vous remercions pour votre offre de collaboration. D'autre part, depuis ce numéro, tous les textes passent par quatre correcteurs orthographiques et grammaticaux: WORDSTAR 2000, WEBSTER COMBO de Simon & Schuster, Hugo Plus et GramR. Vous allez pouvoir ainsi juger par vous-mêmes de leur efficacité.

En ce qui concerne le texte « Dynamique Chaotique » du n°9, il a été retourné pour correction chez l'auteur. La relecture a pris plusieurs mois et, visiblement (pour des diverses raisons), n'a pas résolu le problème. Le tableau de la page 16 est correct et correspond à l'original, il ne comporte aucune erreur. Le signe de la racine carrée amputé de sa barre horizontale vient de XPRESS MAC, sur lequel nous effectuons la mise en page, une autre forme de ce signe, simplement, n'existe pas dans XPRESS...

DISQUETTES ET ANCIENS NUMEROS

☞ ***« Je viens de découvrir votre revue et je suis très agréablement surpris par son niveau technique. (...)***
- N'est-il pas possible de choisir le format des disquettes (3" ou 5")? Disposant d'un lecteur 3", il m'est toujours fastidieux de reconvertir une disquette 5".
- Est-il possible de recevoir vos anciens numéros (1 à 8) ? ».
D. Rollet, Lusigny

Pour l'instant, nous ne pouvons fabriquer que des disquettes au format DFDD 5"1/4 IBM PC - nous n'avons pas de station de duplication équipée en lecteur 3"1/4. Votre problème sera vraisemblablement résolu par notre service télématique 3615 DEBUG, permettant, notamment, le télechargement de programmes. Cette solution sera beaucoup plus élégante, rapide, facile à utiliser et moderne, que le postage des disquettes, qui prend toujours du retard, pose des problèmes de fiabilité et de format.

D'autre part, nous avons encore tous les numéros, sauf le n°1 (voir le bon de commande encarté au centre). Le numéro 1 n'est plus disponible qu'en photocopie...

LISTING N° 9

☞ ***« Je vous écris pour vous demander s'il est possible de recevoir le listing du programme SPA, dont le commentaire est publié dans DEBUG n° 9 sous le titre de « Débogage d'un flot de données séries en Turbo Pascal » par Craig A. Lindley ».***
Ludovic Leilhac. Granville

Dans le n° 9, nous n'avons pas pu insérer les listings de « Un analyseur de protocole de transmission série » et « La Boîte à Communiquer du Macintosh » à cause de leur très grande longueur. Vous trouverez celui de « Un Analyseur de Protocole de Transmission Série » dans notre Cahier Listing de ce mois-çi et celui de « La Boîte à Communiquer du Macintosh » intégralement sur la disquette Listing n° 9.

OPINION

☞ ***« J'ai acheté la version 6.0 de Turbo Pascal. Et bien, vous me croirez si vous voulez, mais le nouvel environnement de l'éditeur est beaucoup moins facile à utiliser que celui de la version 5. La fonction « Delete Watch » est très dure à utiliser, la fonction « Pick Eile » ne me donne pas satisfaction, quant à « Search and Replace », il faut la reconfigurer à chaque fois qu'on s'en sert, c'est vous dire !***
Jean Pierre Foucault. Villejuif

Donc note...

RUMEURS

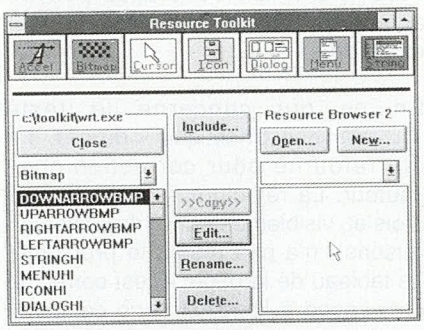
Tendances et Aspects de L'industrie

*par *ART**

☞ **La police de San José,** CA, sera la première à utiliser les ordinateurs portables Gridpad, basés sur la technologie PenPoint. L'ordinateur, dont le clavier est remplacé par un écran sensitif à un crayon, sait lire et interpréter l'écriture humaine. Le système doit réduire le temps de travail du policier, en l'aidant a créer rapidement les rapports de fin de la journée. Actuellement, à San Jose, 1000 policiers créent quelques 100.000 rapports annuellement et un rapport peut être retardé jusqu'à 5 jours avant d'être transmis vers l'investigation. PenPoint, un nouveau système opérationnel 32 bits orienté crayon, présenté par Go Corp de Foster City, CA, est considéré comme une révolution majeure. En éliminant le clavier, PenPoint rend la manipulation du système beaucoup plus naturelle, notamment pour les utilisateurs allergiques au clavier. Intégralement graphique, PenPoint reconnaît 11 gestes et symboles qui commandent les menus, icônes et actions. L'intelligence contextuelle du système permet par exemple de distinguer un cercle dans un dessin, un cercle utilisée comme la lettre "O" et un cercle utilisée dans le traitement de texte pour indiquant le bloc à éditer. Le système est en train d'être livré chez les OEM (IBM, Grid Systems et NCR), les premières réalisations commerciales sont prévues pour la fin de l'année. Alain Lefkof, président de Grid estime qu'un système basé sur un 386 avec 4Mo de RAM pourra coûter de 4 à 7 K$. Parallèlement, Pensoft Corp de San Mateo annonce une application prototype - Gestionnaire Personnel d'Informations. Slate Corp de Scottsdale, Ariz., annonce PenApps, un système de développement pour PenPoint. Selon Dan Bricklin, président de Slate, le portage des applications existantes vers PenPoint « est un changement fondamental. Vous devez vous asseoir et repenser intégralement comment l'application est construite ». On apprend aussi que Bill Campbell quitte Claris pour devenir président et CEO de GO Corp.

Parallèlement, pour contrecarrer, Microsoft a présenté une maquette d'un logiciel concurrent, PenWindow, une extension de Windows 3.1, aussi inexistant. La date de disponibilité de PenWindow est promise pour fin 1991.

☞ **BORLAND** annonce OBJECTVISION version 1.1, un outil de développement pour les utilisateurs désirant de créer sans programmer des applications sous WINDOWS Microsoft. Utilisant extensivement des symboles graphiques, OBJECTVISION, à la limite entre une base de données, tableur et gestionnaire de formulaires, permet de définir des bases de données et gérer les fichiers relatifs. Le prix n'est pas connu au moment de notre parution. Une version Windows de QuattroPro est attendue avant la fin de l'année, la version bêta vient d'être livrée chez les testeurs. Une nouvelle version de C++ compatible Windows à prix réduit (de l'ordre de 99$) est aussi attendue, éliminant la nécessité d'acheter le SDK de chez Microsoft. Le nouveau C++ englobe Whitewater Ressource Toolkit 3.0 (développé avec ACTOR 3.0), un outil permettant de créer visuellement les éléments d'interface comme boites de dialogue, bitmaps, curseurs, icônes et menus interactifs. « L'idée de programmation visuelle est un part important de notre stratégie de bases de données et tableurs » dit Philippe Kahn.



Whitewater Ressource Toolkit

☞ **MICROSOFT** prépare OS/2 version 3.0, appelée aussi « Portable OS/2 ». Cette nouvelle version doit fonctionner sur les processeurs INTEL et RISC et intégrer la compatibilité WINDOWS 3.0. En attendant, l'incertitude sur le futur d'OS/2 continue et les utilisateurs sont de plus en plus perplexes sur les choix à venir. Selon la SPA, les ventes des applications OS/2 en 1990 ont atteint 1.3% du marché. Microsoft annonce aussi que 40.000 ensembles de développement Windows ont été vendus, ce qui promet une pléthore des nouveaux produits Windows dans 12 mois à venir. Un service d'assistance gratuite est ouvert aux développeurs utilisant Windows SDK au 19/1 (206) 454-2030. Il y a aussi des bruits sur l'achat de Zortech par Microsoft, qui serait intéressé par ZC++. Microsoft France, première filiale de Microsoft dans le monde, annonce le franchissement d'un milliard de francs de CA en 1990, soit la croissance de 77%.

☞ Les fabricants de modems américains **Hayes Inc** et **Prometheus** Inc ont conclu une transaction « à l'amiable » terminant le procès intenté par Hayes à Prometheus. Hayes, le fabricant leader de modems, a accusé Prometheus de violer la patente sur la séquence Escape, appelé Heatherington 302, rencontrée nécessairement dans le jeu de commandes de tout modem dit « compatible Hayes ». Hayes a gagné récemment des dommages de 3.5M$ à l'encontre de VanTel, OmniTel et Everex. La patente Hayes est contestée par Multitech Inc. De résultat de ce jugement, attendu par tous les fabricants de modems, dépendent les suites d'autres procès et l'avenir des modems dits « compatibles Hayes ».

☞ **Le juge Terrence J Hatter** a refusé de reconsidérer son verdict du 13 décembre 1990 privant ASHTON TATE de droits d'auteur sur dBase. AT entend de formuler appel devant la 9 éme Cour d'Appel de Los Angeles, ou la procédure prenne normalement 18 mois. Pendant ce temps, AT annonce encore une perte sur son exercice 1990, de 18.1 millions de $, sur les ventes de 230.5 M$.

☞ Après le lancement **d'Excell 3.0** par Microsoft, Lotus promet pour son 1-2-3 Windows à venir « des fonctionnalités jamais auparavant vus dans des tableurs ». En attendant, Lotus a annoncé une perte de 31.6M$, sur les ventes de 186.4M$ de dernier trimestre 1990. C'est une première fois dans son histoire que Lotus annonce des pertes.

☞ **Enrico Pesatori,** 50 ans, l'ancien directeur de la division américaine d'Olivetti, a été nommé président et CEO de Zenith Data Systems. Son objectif sera le développement du marché domestique nord-américain qui, selon lui, serait « sous-exploité depuis un an ». Rappelons que depuis l'acquisition par Bull, ZDS a perdu des énormes contrats fédéraux et que le scandale Mela Associates, qui accusait ZDS de « voler » ses clients gouvernementaux, s'est récemment terminé « hors des tribunaux » par le paiement par ZDS de 250 M$ de dommages à Mela.

☞ **UNISYS** annonce trois contrats à l'Est : deux en Pologne avec la polonaise PKO et les lignes aériennes LOT et un autre, avec la banque de Tchécoslovaquie, qui devient le plus important marché pour l'informatique à l'Est.

☞ **INTEL** annonce des résultats record de 663 M$ sur les ventes de 3.9 milliards de $ ce qui représente 25% de ventes en plus et 66% de profits en plus par rapport à l'exercice 1989. Des nouveaux annonces définissent sa nouvelle stratégie commerciale. Pour combattre la concurrence notamment d'AMD, qui commence à livrer des 386DX et bientôt 386SX, INTEL annonce des nouvelles i486, a prix réduit. Désormais, i486 va exister dans 4 configurations : P23/1 sans FPU, P23/2 sans cache et P24 sans FPU mais avec davantage de cache, tous modèles à des vitesses horloge réduites. Les prix doivent varier de 400 à 500$ en quantité de 1000, comparativement à 670 à

[SERVICE LECTEURS : INTERESSANT = 16 / MOYEN = 17 / MAUVAIS = 18]

860$ actuellement. Certains mauvais esprits prétendent que ces versions ne sont que des exemplaires qui ont failli aux divers tests internes de fabrication... Des informations nous sont parvenus aussi au sujet de i586, un nouveau processeur (nom de code P5) à 66MHz, ayant une architecture interne de 64bits, 16Ko de cache interne (8k pour données et 8k pour les instructions), pourvu des unités d'instructions multiples ce qui permet de travailler en parallèle et exécuter multiples instructions par cycle, selon la technique connue sous le nom de superscaller. Le i586 sera pourvu des nouveaux fonctionnalités de test et de diagnostique : Probe Mode et Performance Monitor. INTEL confirme que le nouveau processeur sera disponible l'année prochaine. Au cours d'ISSCC (International Solid State Circuits Conference), Intel a présenté aussi un « papier » sur un i486 prototype fonctionnant à 100 MHz, totalement opérationnel, qui serait le plus rapide microprocesseur du monde. INTEL veut ainsi montrer qu'il possède la technologie pour atteindre l'objectif fixé pour l'an 2000, à savoir la fabrication d'un microprocesseur de 100 millions de transistors fonctionnant à 250 MHz.

☞ Lors de NetWorld à Boston, **Novell** a annoncé Netware 3.11, une mise à jour importante de Portable Netware 3. On remarque notamment l'abandon de préfixe « 386 », désignant la plate-forme. La nouvelle version est présentée dans les ensembles pour 20, 100 et 250 utilisateurs au prix de 3495$, 6995$ et 12495$ respectivement. La nouvelle version supporte une pléthore des stations de travail sous UNIX (notamment RS/6000), TCP/IP, OS/2 ver 1.3, DOS et Mac. Une version pour AS/400 IBM serait aussi présentée.

☞ Les remous chez **SICOB :** les expositions 1991 ont été annulées, une nouvelle date tentative a été fixée pour l'automne 1992. Parallèlement, SICOB Moscou change de date et est annoncé de 2 à 7 septembre 1991.

☞ Le dernier **PC FORUM** a accueilli, selon ses organisateurs, 55.000 visiteurs. Pour certains exposants, ayant payé pour des stands jusqu'à 2 MFrs, la facture est difficilement digestible : 35 Frs par un contact « virtuel ». L'ambiance morose, dûe à la guerre du Golfe, a eu chance d'être interrompue une fois, lors de la soirée des Micros d'Or. Le grand malaise régnait à la porte, gardée par un service d'ordre musclé : plusieurs personnalités ayant reçu des invitations pour la cérémonie ont été refusés à l'entrée (notamment le Président d'IBM etc.), car ils n'avaient pas le « bon » carton. Le leur invitait simplement à « une soirée privée Micro d'Or », « en tenue cocktail », personne n'a compris la différence. Votre reporter a réussi de se faufiler entre les palmiers. Comme d'habitude, Microsoft, Compaq, Toshiba et Borland ont remporté des prix ce qui a provoqué dans les rangs des cris « Toujours les mêmes! », « On s'ennuie! »,« Montrez-nous des danseuses nues! » etc. Bill Gates,« élu » la personnalité de l'année, n'était pas présent. Le buffet était parfait et somptueux, la fête s'est achevée dans un océan de whisky. On a vu certains confondre des clients et fournisseurs, la porte s'ouvrir mais les danseuses ne sont pas venues. A l'année prochaine!

☞ **PCTechnologie** invite les développeurs Nantucket Clipper à une conférence à Arles de 12 à 14 mars 1990. Tél (1) 42040088 fax (1) 42046980.

☞ **SPA** annonce Steve Jobs, Bernard Vergnes et Jim Manzi comme des intervenants à la deuxième Conférence Européenne à Cannes, qui va débuter le 10 juin prochain. Contactez SPA au (1) 46922703, fax (1) 46922531.

☞ **WANG** annonce Open/Image pour Windows et pour Netware. Open/Image est un ensemble des outils logiciels destinés aux développeurs désirant d'intégrer l'image dans leurs applications. Il apporte quelque 200 fonctions permettant de manipuler l'image sous Windows et Netware. Lors du dernier PC Forum, WANG a présenté aussi ses micro-ordinateurs basés sur des processeurs Intel, bus ISA et MCA.

Soirée PC Forum 1991

Les organisateurs du 8e PC Forum

prient Monsieur [redacted]

de leur faire le plaisir de participer
à la première soirée privée PC Forum,
le mercredi 13 février à 19 heures 30 précises
au Parc des Expositions de Paris,
Porte de Versailles.

Tenue de Cocktail
Accès parking porte E
Voiture avec chauffeur, rentrer par la porte A

Cette invitation, strictement personnelle, valable pour deux personnes, sera exigée à l'entrée.

L'os de discorde : le carton d'invitation PC. FORUM

PC 480/25 C de Wang

☞ **Le portable IBM 386SX,** IBM à 20Mhz de 3.5 Kg, avec des disques durs de 30,40 ou 60Mo, 2Mo de mémoire, VGA et modem 2400b sera retardé du aux problèmes d'approvisionnement en processeurs et disques durs. Le coût annoncé aux USA est de l'ordre de 6000$, pour la configuration avec disque 60Mo. parallélement, IBM a annoncé deux nouvelles machines dans la ligne PS/2 : modèle 90 et 95. Basées sur i486, équipés de la carte graphique en mode XGA native et adaptateurs SCSI, ces deux machines MCA (une « tour » et un « desktop ») sont les premières réalisations IBM adaptés vraiment à ce processeur. La nouveauté : la machine n'est plus construite sur monocarte. Sur la carte mère il n'y a que les connecteurs MCA, adaptateurs série, parallèle, souris, clavier, vidéo et contrôleur du disque souple (un 3"1/2 et un 5"1/4). Le processeur et tous les autres circuits importants MCA sont implémentés sur une carte spécifique, appelé ProcessorComplex. Les prix eu USA commencent à 14145$ (avec disque de 320Mo) à 17745$.

☞ Un nouveau utilitaire de **Peter Norton Computing Inc.** est annoncé par SYMANTEC. The Norton AntiVirus, comme le nom le suggère, est destiné à « détecter, éradiquer et protéger les systèmes sous DOS de tout type de virus ». Le logiciel apporte aussi le verrouillage d'accès à l'ordinateur par un mot de passe. le prix en USA est de 129.95 $. Symantec annonce aussi une nouvelle version 2.0 d'Editeur Norton, apportant un environnement à fenêtres multiples et la souris.

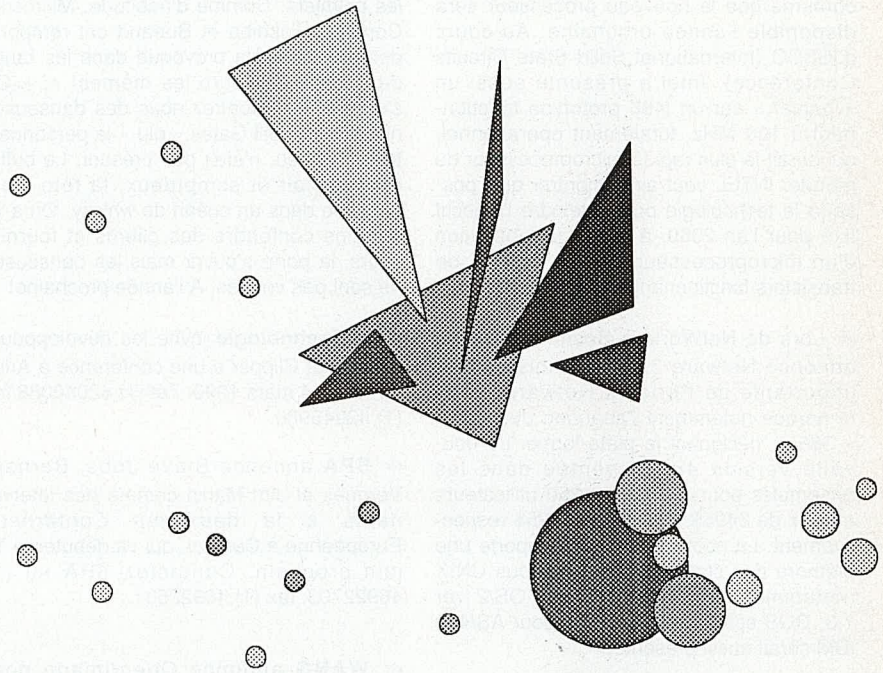
☞ **GOTO Informatique** annonce une nouvelle version de Wellcom, un logiciel de télécommunication pour PC/AT/PS. La nouvelle version, appelée OVERCOM, apporte l'émulation Minitel, TTY, VT52 et VT100, reconnaît les protocoles Kermit, X et Y modem, un langage de commandes et la souris. Livrée avec un manuel, câble Minitel et l'adaptateur DB9/DB25 Overcom coûte 750 Frs HT.

☞ **INNOSOFT** annonce TSR Made Easy, une bibliothèque permettant de développer des logiciels résidants en Turbo Pascal 6.0. Prix : 890 Frs HT, tél : (1) 45.06.76.91, fax : (1) 47.28.62.89.

LA PROGRAMMATION ORIENTÉE

La Programmation Orientée Objet (POO) peut aussi bien s'appliquer à l'assembleur qu'aux langages de haut niveau.

Par Randall L.Hyde



L'un des buts de la programmation orientée objet est de simplifier l' écriture de nombreux types de programmes. La programmation orientée objet n'est cependant pas la panacée. C'est une technique, comme la récursivité, que l'on peut utiliser parfois pour se sortir de certains problèmes de programmation. Dans des cas bien précis, il est préférable de recourir aux objets, mais il en existe de très nombreux pour lesquels la programmation orientée objet n'apporte rien. Néanmoins, les techniques de programmation orientée objet sont un précieux outil que chacun doit avoir dans sa besace.

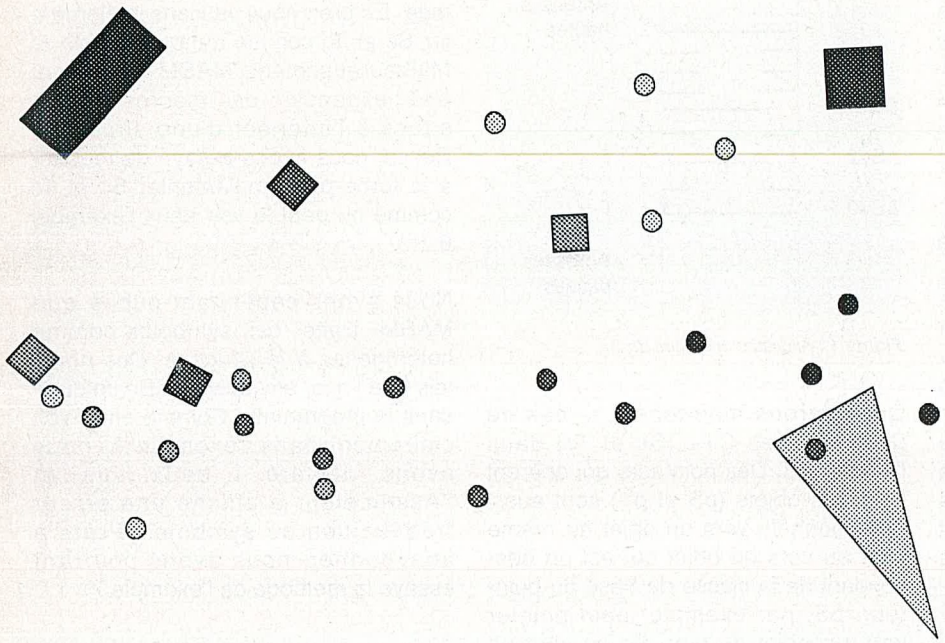
La POO n'est pas nouvelle. On la rencontre ici ou là depuis la fin des années 60. Et pourtant, elle se languissait dans des laboratoires jusqu'à ce que le Pascal Objet et le C++ réussissant leur opération de charme. Le succès de ces langages a démontré que la POO n'est pas le domaine réservé de quelques langages de programmation ésotériques. On peut, bien au contraire, appliquer la POO à presque n'importe quel langage de programmation.

On peut cependant penser que le langage Assembleur n'est pas construit pour la programmation orientée objet. Mais n'oublions pas que l'on disait la même chose du Pascal et du C, il y a cinq ans.

A quoi ressemble un programme orienté objet écrit en Assembleur ? Ou plutôt : "Quelle est l'essence d'un programme orienté objet et comment l'introduit-on dans un programme Assembleur ?" Une fois que l'on a débarassé de l'habillage et des conventions de notation d'un langage type C++, on constate que les deux traits saillants d'un programme orienté objet se nomment polymorphisme et héritage.

[SERVICE LECTEURS : INTERESSANT = 19 / MOYEN = 20 / MAUVAIS = 21]

OBJET EN ASSEMBLEUR



Dans la plupart des langages de programmation, le polymorphisme se divise en deux formes principales : la forme statique et la forme dynamique. L'idée de base, cependant, est la même : on appelle des sous-routines différentes par le même nom. Afin de savoir quelle routine appeler le polymorphisme statique permet des conventions de notations de la forme surcharge opérateur/fonction dans des langages tel que le C++. Le polymorphisme statique utilise la liste des paramètres conjointement au nom de la routine (ils forment, ensemble, la signature de la routine), Par exemple, examinons les routines C suivantes :

```
CmplxAddCC(C1, C2, C3) ;

                /* C1 = C2 + C3 ; */

CmplxAddCR(C1, C2, R1) ;

                /* C1 = C2 + ToCmplx(R1) ; */

CmplxAddRC(C1, R1, C2) ;

                /* C1 = ToCmplx(R1) + C2 ; */
```

Ces fonctions additionnent des nombres complexes et réels. Il faut une fonction différente pour chaque type d'addition. Par contre, en C++ on écrirait :

```
CmplxAdd(C1, C2, C3) ;
CmplxAdd(C1, C2, R1) ;
CmplxAdd(C1, R1, C2) ;
```

Le compilateur C++ détermine s'il faut appeler CmplxAddCC, CmplxAddCR ou CmplxAddRC. En fait, il serait également possible de surcharger l'opérateur "+" du C++ et d'utiliser les trois formes C1=C2+C3;, C1=C2+R1; ou C1=R1+C2;, mais l'exemple ci-dessus resterait valable.

La surcharge statique, bien que pratique, n'apporte aucune puissance au langage. Les appels à CmplxAdd

Randall L. Hyde est le concepteur de nombreux projets logiciels et matériels, dont des assembleurs pour divers systèmes. En plus de son activité de consultant, il est professeur d'informatique à l'université polytechnique de Californie, et à UC Riverside. On peut le contacter à 9570 Calle La Cuesta, Riverside, CA 92503, USA.

indivisent trois routines différentes. CmplxAdd (C1,C2,C3) appelle CmplxAddCC, CmplxAdd (C1,C2,R1) appelle CmplxAddCR et CmplxAdd (C1,R1,C2) appelle CmplxAddRC. Le compilateur C++ détermine la routine appelable lors de la compilation. Le polymorphisme statique est un mécanisme qui permet au compilateur de choisir parmi d'autres la routine relative à la signature de l'appel de la fonction.

Il est aussi possible d'assigner la même signature à l'appel de plusieurs routines. Supposons, par exemple, que vous ayez une classe Forme comprenant trois objets graphiques : des cercles, des rectangles et des triangles. Si vous disposez d'un objet arbitraire du type Forme, le compilateur ne peut pas déterminer quelle routine DESSIN appeler. Le programme la définit au moment de l'exécution. Cela permet à un appel unique de dessiner des cercles, des rectangles et des triangles, au moment de l'exécution, avec les mêmes instructions machine. C'est le polymorphisme dynamique : la détermination, au moment de l'exécution, de la routine à appeler. Le C++ utilise des fonctions virtuelles et le Pascal orienté objet met en jeu des fonctions et des procédures surchargées pour implémenter le polymorphisme dynamique.

L'héritage permet de construire des sur-ensembles de structures de données à partir des structures existantes. Ce mécanisme rend possible la généralisation des types de données, autorisant la manipulation d'objets divers indépendamment de leur type. On peut, alors, définir des formes aussi diverses que le cercle, le rectangle et le triangle et les traiter comme des structures compatibles.

L'implémentation des classes et de l'héritages

A cause de la proximité des notions de structure et de classe, il peut être pertinent d'examiner l'implémentation des structures avant d'étudier les classes. Soit S, une variable du type décrit dans l'exemple 1.

```
struct
{
        int i;
        int j;
        char *c;
} S;
```

Exemple 1 : *La variable S*

Le compilateur a besoin de réserver, quelque part en mémoire, un emplacement de stockage pour les champs de S. Habituellement, les compilateurs placent ces champs côte à côte (voir la figure 1). L'Assembleur de Microsoft (MASM) déclare des structures de manière similaire, comme on peut le voir dans l'exemple 2. Si S représente l'adresse de base de cette structure, S+0 est l'adresse de S.i, S+2 est l'adresse de S.j et S+4 est l'adresse de S.c.

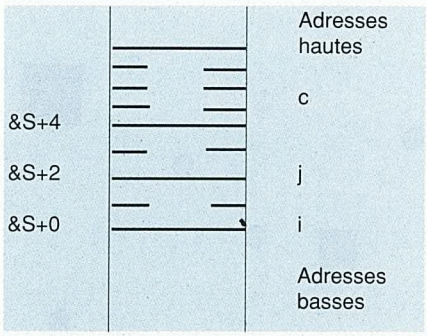


Figure 1 : *Allocation mémoire de S*

Considérons maintenant le cas de deux classes C++ (Sc et Tc) dans l'exemple 3. Des pointeurs qui opèrent vers des objets (pS et pT) sont aussi opérationnels vers un objet du même type ou vers un objet qui est un descendant de la classe de base du pointeur. pS, par exemple, peut pointer vers un objet de type Sc ou vers un objet de type Tc. Souvenez-vous que l'expression *pS.j est équivalente à (int)*(pS+2); par conséquent, si pS pointe vers un objet de type Tc, le champ j doit également apparaître avec à l'offset deux à l'intérieur de la structure. Pour un fonctionnement correct de l'héritage, les champs communs doivent apparaître avec le même décalage à l'intérieur de la structure (voir figure 2).

Les champs additionnels d'une sous-classe apparaissent souvent après les champs de la classe parent, c'est pourquoi la majorité des compilateurs implémentent la classe Tc comme dans l'exemple 4. Remarquez que les décalages de i, j et c sont les mêmes pour Sc et Tc.

En explorant la manière d'implémenter l'héritage en Assembleur, nous avons eu la bonne idée d'utiliser des macros à l'intérieur des définitions de structure afin de résoudre le problème de l'héritage. En bref, nous voulions implémenter Sc et Tc comme dans l'exemple 5. Malheureusement, MASM ne permet pas l'expansion des macros ou des strucs à l'intérieur d'une structure. Déçus, nous avons essayé de recourir à la force pour implémenter Sc et Tc comme on peut le voir dans l'exemple 6.

Nous avons cependant oublié que MASM traite ces symboles comme hétérogènes à la structure. Des noms tels que i, j, c, etc..doivent être uniques dans le programme. Comme on le voit clairement dans l'exemple 6, nous avons déclaré i deux fois et l'Assembleur a affiché une erreur "redéfinition de symbole". Prêts à abandonner, nous avons pourtant essayé la méthode de l'exemple 7.

MASM affect , à l'intérieur de la structure, les offsets aux noms des champs. Ainsi, i reçoit la valeur zéro, j reçoit deux, etc. MASM n'associe pas i aux étiquettes de type Sc. Vous pouvez adopter les symboles T.j et S.j dans votre programme. Passer que l'opérateur "." se comporte comme l'opérateur "+", T.j est équivalent à T+2.Pour que Tc puisse hériter des champs de Sc, tout ce que nous avons à faire est de

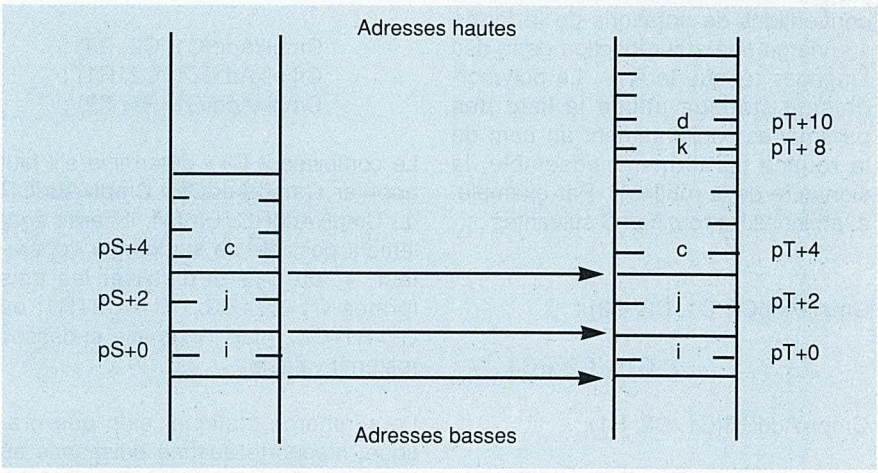


Figure 2 : *Allocation mémoire de *pT et *pS*

```
SType struct
i       dw      ?
j       dw      ?
c       dd      ?
SType ends
```

Exemple 2 : *La déclaration de structure avec MASM*

réserver suffisamment de place au début de la structure Tc pour chacun des champs de Sc. Nous avons inséré au début deux pseudo-codes DW et un DD pour réserver l'espace nécessaire aux champs i, j et c.

Cette technique pourrait être gênante si le nombre de champs hérités devait s'élever notablement. Le programme de l'exemple 8 résout ce problème. Le premier pseudo-code DB dans Tc réserve l'emplacement idoine à l'héritage de tous les champs de Sc par TC. De même, Uc (une sous - classe de Tc) réserve l'espace nécessaire en début de structure pour les champs hérités de Tc et Sc. Le programme de l'exemple 8 fonctionne parfaitement quand vous n'avez besoin d'initialiser aucun des champs hérités de Sc ; si au contraire vous avez besoin d'initialiser certains de ces champs, il vous faudra employer une méthode drastique et déclarer à nouveau un emplacement pour chaque champ.

Les méthodes

Jusqu'à maintenant, nous avons traité de l'implémentation d'objets dont les champs sont tous des variables. Que se passe t-il lorsque l'on introduit des méthodes ? A condition de ne pas surcharger une méthode, il est possible de la traiter comme que toute autre procédure Assembleur et de l'appeler directement. Si une méthode est surchargée, elle doit être appelée grâce à un pointeur à l'intérieur de l'objet. Considérons la déclaration en C++ de la classe de l'exemple 9. Le code Assembleur correspondant à cette classe est donné dans l'exemple 10. Pour appeler S.geti, on utilise l'instruction du 8086 : CALL S.geti.

Du fait que S.geti est une variable de type double mot, l'instruction CALL appelle la procédure S.geti pointant vers Sc_geti. L'appel indirect des méthodes nous sera utile lorsque nous examinerons plus loin la surcharge des méthodes .

THIS

Supposons que nous ayons trois instances de la classe Sc, soit S1, S2 et S3, déclarées en Assembleur de manière suivante :

```
S1  Sc
S2  Sc
S3  Sc
```

S1.geti, S2.geti et S3.geti appellent tous la même procédure (disons

```
class Sc
{
        int i;
        int j;

        char *c;
};

Sc *pS;
Tc *pT;
```

```
 TCclass Tc:Sc
{
        int k;
        char *d;
};
```

Exemple 3 : *Classes C++*

Sc_geti). Comment Sc_geti établit-il une différence entre S1.i, S2.i et S3.i ? Dans les langages orientés objet, type Pascal objet ou C++, le compilateur transmet automatiquement à la méthode un paramètre spécial appellé this. This pointe toujours vers l'objet utilisé pour appeler la méthode. Lorsque l'on exécute S1.geti, le compilateur transfère l'adresse de S1 dans this vers geti. Le compilateur traite de la même manière l'adresse de S2 dans this quand on appelle S2.geti.

On assigne this à une méthode comme on le ferait avec n'importe quel paramètre. Nous avons adopté la convention de transférer this dans les registres ES:BX parce que la meilleure façon de traiter des paramètres est de les introduire dans les registres du 8086. La méthode Sc_geti doit ressembler à l'exemple 11 - à condition que nous renvoyons i dans le registre AX.

Cet exemple nous montre un des problèmes fondamentaux de la programmation orientée objet : son inefficacité. L'exemple 12 charge S1.i dans AX. Il faut six instructions alors que, logiquement, on devrait s'en contenter d'une : mov ax, S1.i. Bienvenue dans le monde merveilleux de la programmation orientée objet ! Et pourtant supprimer tous ces éléments supplémentaires, en chargeant S1.i directement en AX, annulerait les avantages de la programmation orientée objet.

En fait, la situation n'est pas aussi mauvaise qu'elle paraît.Généralement, ES:BX pointera déjà vers l'objet auquel on veut accéder. Il n'en demeure pas moins que l'appel et le retour constituent des charges très lourdes en cas de chargement du registre AX avec la valeur d'un mot. Stroustrup avait prévu cette difficulté , lorsqu'il conçut le langage C++. Il la résolut en introduisant des fonctions en ligne (des macros). On peut utiliser cette même technique en Assembleur dans un but d'optimisation comme le montre l'exemple 13. Ce petit programme introduit une autre convention qui nous est chère : l'utilisation des macros pour tous les appels de méthode, même ceux qui sont de vrais appels de fonctions. Cela permet l'usage d'un format d'appel cohérent pour toutes les méthodes : de vraies fonctions ou macros.

Ces macros possèdent un inconvénient central : impossibilité de surcharger les procédures. Ce défaut existe également pour les fonctions en ligne du langage C++. On ne peut pas avoir une fonction virtuelle en ligne. Il ne faut, par conséquent, utiliser cette technique que pour des méthodes bien précises que vous n'aurez jamais à les surcharger. Heureusement, l'implémentation de la macro facilitera son remplacement par un appel de fonction si besoin est de surcharger la procédure. Il suffit de substituer le code en ligne par un appel à l'intérieur de la macro.

```
struct Tc               Tc      struct
{
        int i;          i       dw      ?
        int j;          j       dw      ?
        char *c;        c       dd      ?
        int k;          k       dw      ?
        char *d         d       dd      ?
};                      Tc      ends
```

Exemple 4 : *La manière de la plupart des compilateurs d'implémenter une classe comme Tc*

Polymorphisme et surcharge

Les procédures de surcharge permettent à la "même" méthode d'effectuer des opérations différentes selon l'objet qui lui est transmis. Examinons les définitions de classe de l'exemple 14. Rect et Cercle sont des types dérivés de Forme. Si ES:BX pointe vers une forme générique (c'est-à-dire que ES:BX pointe vers un objet de type Forme, Rect ou Cercle) alors CALL _THIS.Dessin appellera Dessin_Forme, Dessin_Rect ou Dessin_Cercle, selon l'objet pointé par ES:BX. Cela permet d'écrire du code générique dispensé de connaître les détails de la forme qu'il est en train de dessiner. L'objet sait lui-même se servir du pointeur vers la routine de dessin spécifique pour se dessiner.

```
ObjetsSc macro
i        dw      ?
j        dw      ?
c        dd      ?
         endm

;
ObjetsTc macro
         ObjetsSc
k        dw      ?
d        dd      ?
endm

;
Sc       struct
         ObjetsSc
Sc       ends

Tc       struct
         ObjetsTc
Tc       ends
```

Exemple 5 : *Une approche d'implémentation de Sc et Tc*

Attribution d'objets

Les langages (orientés objet) évolués tels que le Pascal objet et le C++ ont tendance à cacher de nombreux détails de l'attribution. En Assembleur, bien entendu, le programmeur doit opérer toute l'attribution à la main. Bien qu'une discussion approfondie sur l'attribution dynamique des objets n'entre pas dans le cadre de cet article, le sujet est si répandu qu'il mérite qu'on le mentionne brièvement.

L'attribution statique d'un objet en Assembleur est très simple. Si l'on reprend les définitions de la classe Forme (Forme, Rect et Cercle)

```
Sc       struct
i        dw      ?
j        dw      ?
c        dd      ?
Sc       ends

Tc       struct
i        dw      ?
j        dw      ?
c        dd      ?
k        dw      ?
d        dd      ?
Tc       ends
```

Exemple 6 : *La méthode brutale pour implémenter Sc et Tc*

données ci-dessus, on peut aisément proposer des variables de type correspondant moyennant des assertions de la forme :

```
                MonRect
Rect
                MonCercle
Cercle
                MaForme
Forme
```

Cela remplit automatiquement le champ DESSIN de ces variables (le linker/loader remplit de telles adresses lorsqu'il charge le programme en mémoire). Que se passe-t-il si l'on attribue dynamiquement un objet ? Supposons que nous disposions d'une routine, alloc, à laquelle nous transmettons un nombre d'octets dans CX, et qu'elle retourne un pointeur vers un bloc mémoire de cette taille dans ES:BX. Attribuons un objet de type rectangle comme dans l'exemple 15. Alloc ne sera pas assez aimable pour remplir le pointeur du champ Dessin_Rect. Nous devons faire cela nous -mêmes, et cela nécessite les quatre instructions de l'exemple 16.

```
Sc       struct
i        dw      ?
j        dw      ?
c        dd      ?
Sc       ends

Tc       struct
         dw      ?
         dw      ?
         dd      ?
k        dw      ?
d        dd      ?
Tc       ends

S        Sc
T        Tc
```

Exemple 7 : *Encore une tentative d'implémenter Sc et Tc*

Huit instructions : cela peut paraître insuffisant pour créer un simple objet. Souvenons-nous, cependant, que notre objet simple Forme n'a qu'une méthode surchargée. S'il y avait une douzaine de méthodes, il faudrait 52 instructions. Il devient clair qu'une procédure CREE commence à avoir un sens. Chaque sous-classe (Forme, Rect et Cercle) aura besoin de sa propre méthode CREE. Normalement, on ne surcharge pas une méthode CREE parce que, au cours du processus de création, on sait exactement quel est le type d'objet que l'on crée. Les méthodes CREE que nous écrivons attribuent toujours l'espace mémoire voulu, initialisent tous les champs majeurs puis retournent un pointeur sur le nouvel objet dans ES:BX. Le programme de l'exemple 17 utilise les types Rect et Cercle.

```
Sc       struct
i        dw      ?
j        dw      ?
c        dd      ?
Sc       ends

Tc       struct
         db      (size Sc) dup (?)
k        dw      ?
d        dd      ?
Tc       ends

Uc       struct
         db      (size Tc) dup (?)
e        dd
Uc       ends

S        Sc
T        Tc
U        Uc
```

Exemple 8 : *La bonne manière d'implémenter Sc et Tc*

Nous n'avons besoin, pour manipuler ces objets, que de charger grâce à this le pointeur approprié dans ES:BX et d'accéder aux champs ou aux méthodes souhaitées.

Autres conventions

Nous avons jugé utile, lors de l'écriture de programmes orientés objet en Assembleur, de disposer de lignes directrices pendant les phases d'élaboration (c'est à dire avant de prendre en considération l'efficacité). La plupart de ces lignes de force sont des pratiques largement admises en programmation objet; d'autres appartiennent essentiellement à l'Assembleur.

```
class Sc
{
        int i,j;
        char *c;

        public:

        int geti () {return(i)};                    // en ignorant le fait
        int get j() {return(j)};                    // que C++ implémente
        void seti (x) int x; {i = x;};     // ces méthodes // en ligne.
        void setj (x) int x; {j = x;}
};
```

Exemple 9 : *Une déclaration de classe en C++*

Voici les principales que nous mettons en oeuvre :

. Essayer d'utiliser l'attribution dynamique des objets chaque fois que cela est possible. Dans les programmes orientés objet les mieux construits, les instances d'un objet apparaissent et disparaissent tout au long du programme. Il est rare qu'une même instance existe du début à la fin d'un programme. Parce que les méthodes d'un objet se renvoient toujours indirectement à ses champs , les objets attribués statiquement dynamique des objets chaque fois que cela est possible. Dans les programmes orientés objet les mieux construits, les instances d'un objet apparaissent et disparaissent tout au long du programme. Il est rare qu'une même instance existe du début à la fin d'un programme. Parce que les méthodes d'un objet se renvoient toujours indirectement à ses champs , les objets attribués statiquement présentent peu d'avantages. Convertir, ultérieurement, un objet attribué statiquement en un objet attribué dynamiquement n'est pas une bonne chose. Il vaut mieux agir du premier coup !

. Eviter d'accéder aux variables individuelles (champs) à l'intérieur d'un objet. Ecrire des méthodes qui stockent des valeurs à l'intérieur de ces champs et récupèrent des valeurs à partir de ces mêmes champs. La technique d'information cachée est bien rôdée en POO et ne mérite pas d'être discutée plus avant.

. Surcharger autant de méthodes que possible. CREE est probablement la seule méthode qui ne doit pas être surchargée. Les méthodes d'accès, permettant d'atteindre champs des classes les plus éloignées, pourraient être un autre motif militant pour l'accès direct. Mais une perte de généralité pour un faible gain d'efficacité que cela signifie en vaut rarement la peine.

. Toujours utiliser des macros pour appeler des méthodes, notamment celles que vous n'appelez pas indirectement. Cela constitue un mécanisme cohérent d'appel des méthodes facilite la surcharge des méthodes que l'on choisit d'implémenter initialement en ligne ou sans surcharge. Cela s'applique tout aussi bien à l'accès aux champs à l'intérieur d'un objet.

Il est bien évident que ce ne sont là que des lignes directrices et non pas des règles absolues . Mais un peu de discipline au début d'un projet est une aide précieuse pour éviter de se disperser ensuite.

```
Sc      struct
i       dw      ?
j       dw      ?
c       dd      ?
geti    dd      Sc_geti
getj    dd      Sc_getj
seti    dd      Sc_seti
setj    dd      Sc_setj
Sc      ends

S       Sc
```

Exemple 10 : *Le code Assembleur correspondant à l'exemple 9*

Un exemple

L'exemple du listing 1 (page xxx) est un programme qui ajoute, soustrait et compare des entiers binaires signés, des nombres entiers non signés et des valeurs décimales codées binaires DCB.

```
_THIS   equ     es:[bx]
Sc_geti proc    far
        mov     ax,_THIS.i
        ret
Sc_geti endp
```

Exemple 11: *La méthode Sc_ geti*

```
mov     bx, seg S1
                mov     es, bx
                mov     bx, offset S1
                call    S1.geti
;En supposant que S1 est dans le segment de données
```

Exemple 12 : *Charger S1.i dans ax*

```
;
; Expansion en ligne de geti pour optimisation
;
_Geti     macro
                    mov
ax, _THIS.i
                    endm
;
; Effectue un appel réel à une fonction
; trop longue pour l'expansion en ligne
;
_Printi   macro
                    call
_THIS.Printi
                    endm
                    .
                    .
                    .
                    _Geti     ;Charge i
dans AX
                    .
                    .
                    .
                    _Printi   ;Appelle routine Printi
```

Exemple 13 : *Optimisation*

Bien qu'il s'agisse d'un exemple incomplet (il lui manque plusieurs méthodes importantes telles que CREE, AFFICHE, DISPOSE etc..), il révèle l'essence de la programmation orientée objet en Assembleur.

```
Forme     struct
sgx                 dw        ?
;Coordonnée supérieure gauche X
sgy                 dw        ?
;Coordonnée supérieure gauche Y
idx                 dw        ?
;Coordonnée inférieure droite X
idy                 dw        ?
;Coordonnée inférieure droite Y
Dessin    dd        Dessin_Forme
;Routine par défaut (surchargée)
Forme     ends
;
Rect                struct
                    dw        4 dup (?)
;Réserve place pour coordonnées
                    dd        Dessin_Rect
;Dessine un rectangle
Rect                ends

Cercle    struct
                    dw        4 dup (?)
;Réserve place pour coordonnées
                    dd
Dessin_Cercle       ;Dessine un cercle
Cercle    ends
```

Exemple 14 : *Définition de classes typiques*

Et vos programmes ?

La programmation orientée objet est un concept qui peut réduire le temps que vous passez à développer certaines catégories de programmes. Le concept POO s'applique aussi bien à l'Assembleur qu'aux programmes en langage évolué. Le seul inconvénient est que l'on ne dispose pas d'un grande bibliothèque de classe permettant de construire nos propres objets. Bien évidemment, ces mêmes problèmes existent pour les utilisateurs de Pascal objet et de C++. Le temps résoudra ce problème pour ces langages, au fur et à mesure que les utilisateurs commenceront à développer des modules réutilisables pour les deux niveaux de langage. Ce vice rédhibitoire empêche l'Assembleur orienté objet de prendre son essor. Il sera peut être possible, un jour, d'acheter des bibliothèques en Assembleur orienté objet en coffret tout prêt.

```
mov       cx, size Rect
call      alloc
mov       word ptr MonRectPtr,   bx
mov       word ptr  MonRectPtr+2, es
```

Exemple 15 : *Code pour allouer un rectangle*

Mais, en attendant , il faudra que vous écriviez les vôtres. Même dans ces conditions, les petites difficultés et les techniques de la programmation orientée objet méritent d'être prises en considération pour votre prochain projet écrit en langage Assembleur.

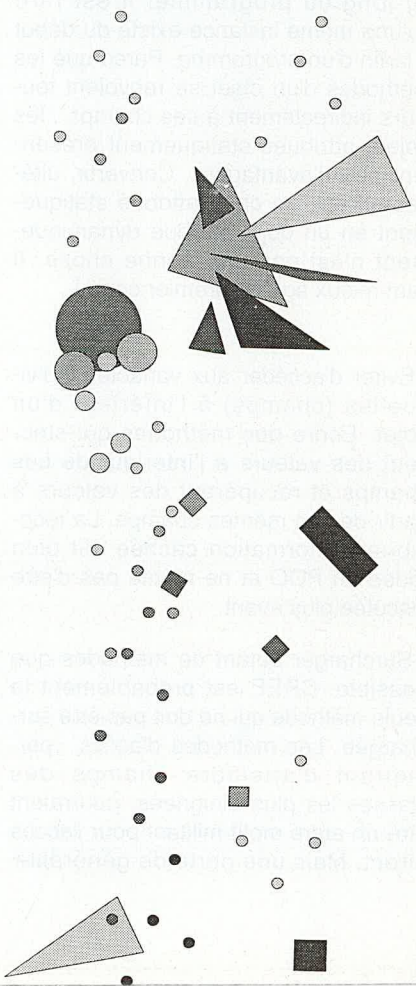
```
mov       ax, offset DESSINRect
mov       _THIS.Dessin, ax
mov       ax, seg DESSINRect
mov       _THIS.Dessin+2, ax
```

Exemple 16 : *Initialisation du pointeur Dessin_Rect*

```
                    mov       cx, size Cercle
                    call      CreeCercle
                    mov       word ptr VarCerclePtr,
bx
                    mov       word ptr
VarCerclePtr+2, es
;
                    mov       cx, size Rect
                    call      CreeRect
                    mov       word ptr VarRectPtr,
bx
                    mov       word ptr VarRectPtr+2,
es
```

Exemple 17 : *Exemple d'utilisation des types Rect et Cercle*

La Programmation Orientée Objet en Assembleur : copyright **Dr. Dobb'S Journal**, M & T Publishing, Inc., Redwood City, CA USA 1990. All right reserved.



19 AU 22 MARS 1991 • 9e CONGRÈS MONDIAL DE LA PROTECTION ET DE LA SÉCURITÉ INFORMATIQUE ET DES COMMUNICATIONS

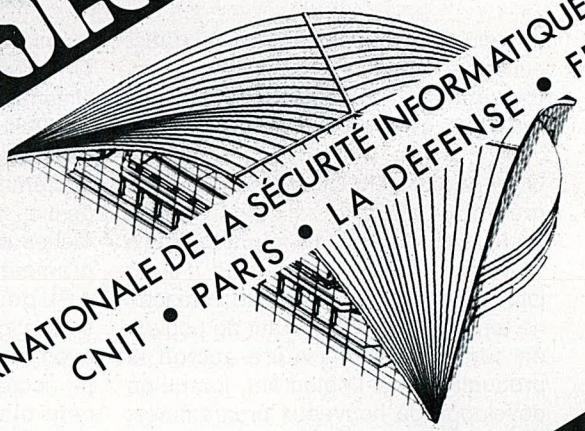
■ SECURICOM est depuis 9 ans le congrès international le plus spectaculaire et le plus efficace en sécurité informatique traitant concrètement les méthodes, expériences et recherches les plus avancées dans ce domaine.

■ Journée Exceptionnelle de Formation le 19 mars sur l'actualité informatique la plus brûlante : « La Sécurisation des Bases de Données Distribuées » par l'expert mondial n° 1, Shaku ATRE (USA) assistée de Stanley KURZBAN (USA).

■ NOUVEAU : Ateliers de sécurité « constructeurs » spécialement conçus pour les utilisateurs des systèmes BULL, DEC et IBM.

Pour recevoir le programme détaillé du Congrès ainsi qu'un bulletin d'inscription à Securicom 91 :
SEDEP-BLENHEIM
22-24, rue du Président Wilson
92532 Levallois-Perret Cedex - France
Tél. : (1) 47 56 50 00
Fax : (1) 47 56 08 18

SECURICOM 91



2e SEMAINE INTERNATIONALE DE LA SÉCURITÉ INFORMATIQUE ET DES COMMUNICATIONS
CNIT • PARIS • LA DÉFENSE • FRANCE

infosec 91

Pour toute information complémentaire sur Infosec 91 :
MCI
8, rue de l'Isly
75008 Paris – France
Tél. : (1) 42 94 27 67
Fax : (1) 42 93 29 67
Télex : 281033

■ Le salon leader de la sécurité informatique en Europe.
■ 150 exposants.
■ La vitrine complète des matériels, équipements et services pour la sécurité informatique : matériels informatiques, matériels connexes à l'informatique, matériels de sécurité informatique, logiciels de sécurité, conseils, assurances, moyens de secours...
■ 6000 visiteurs professionnels attendus.
■ Journée d'initiation en sécurité informatique le 19 mars.

20 AU 22 MARS 1991 • 5e SALON & COMMUNICATIONS SUR LA SÉCURITÉ INFORMATIQUE

SERVICE LECTEURS : INTERESSANT = 4 / MOYEN = 5 / MAUVAIS = 6

KERMIT

1ère Partie

par Brian R. Anderson

La courbe d'apprentissage d'OS/2 peut être raide, mais l'effort sera récompensé.

J'appartiens à cette race d'entêtés du DOS qui renonce à OS/2 depuis son arrivée il y a 3 ans. « Trop cher » dis-je, « trop lent ». Il a besoin de trop de mémoire . Pourquoi un seul utilisateur a-t-il besoin de multi-tâche ? Pourquoi a-t-on besoin d'une mémoire protégée sur un système non partagé ? A posteriori, cette histoire ressemble à celle des raisins verts : je ne disposais pas des moyens pour mettre mon ordinateur individuel à niveau, de manière à le faire tourner sous OS/2.

J'ai acheté, récemment, un nouvel ordinateur avec un 80386 à 33 MHz, un moniteur VGA et un disque dur rapide de 150 Mo. Soudainement, tout est devenu très différent. C'est vrai, OS/2 accapare beaucoup plus du temps système que le DOS mais, à 33 MHz, on ne s'en aperçoit pas vraiment.

C'est vrai, 4 Mo c'est beaucoup de mémoire, mais le prix des RAM a beaucoup baissé. Le multi-tâche a vraiment un sens - le fait de n'avoir jamais besoin d'attendre que la machine termine une tâche avant de pouvoir en commencer une autre accroît la productivité. Et finalement, lorsqu'on développe de nouveaux programmes, une protection contre vos propres programmes qui comportent encore des erreurs est un véritable avantage - cela élimine virtuellement le besoin de réinitialiser le système à la suite d'un incident. Après cette confession et cette rétractation passons au sujet qui nous intéresse.

KERMIT est un protocole de transfert de fichiers qui permet à divers ordinateurs de communiquer. Mon article KERMIT RENCONTRE MODULA-2 du mois de Janvier 1991 (Debug Magazine numéro 9) décrivait le protocole PC KERMIT et ma manière de l'implémenter en MODULA-2. Cet article-ci décrit la manière dont j'ai porté PC KERMIT sur OS/2 Présentation Manager.

OS/2 Mini-Primeur

Operating System 2 et Presentation Manager sont de très gros systèmes. Le noyau d'OS/2 possède plus de 250 fonctions et celui du PM plus de 500. On appelle cet ensemble de fonctions l'Interface de Programme d'Application, ou l'API. Le noyau API vous apporte tous les services traditionnels d'un système d'exploitation, y compris la gestion de la mémoire, le contrôle des tâches et des entrées/sorties. Un programmeur qui n'utilise que le noyau API peut écrire des programmes sophistiqués en mode caractères, sans avoir jamais besoin d'avoir recours à un accès direct au matériel, alors que cela était souvent nécessaire sous MS/DOS.

Le noyau API est divisé en 4 parties : DOS, Kbd, Mou et Vio. Les Kbd, Mou et Vio apportent un accès rapide et flexible au clavier, à la souris et à l'écran, essentiellement utilisés dans les applications en mode caractères. Présentation Manager prend en charge le clavier et la souris de telle manière que le groupe de fonctions Kbd et Mou n'est pas utilisé dans une application sous PM. PM dispose de nombreuses options pour le contrôle de l'écran. Il permet aussi bien les graphiques en mode vectoriel que tramé, ainsi que diverses fontes proportionnelles et non proportionnelles. Il est possible d'utiliser le groupe de fonctions Vio dans une application PM dans une forme légèrement modifiée appelée AVio « Advanced Vio ». Mon implémentation de KERMIT utilise les fonctions Avio pour émuler l'écran d'un terminal standard (Video Display Terminal VDT) et permet par conséquent une interaction avec les gros systèmes.

[SERVICE LECTEURS : INTERESSANT = 22 / MOYEN = 23 / MAUVAIS = 24]

POUR OS/2

La partie DOS du noyau OS/2 gère à la fois les entrées/sorties sur disques, la gestion des tâches et la gestion de la mémoire, pour les applications DOS et les applications sous PM.

L'API de Presentation Manager est également divisée en modules : Dev, Gpi, et Win. Les fonctions du groupe Dev sont utilisées pour accéder aux drivers de périphériques PM, ce qui permet d'avoir une gestion du graphisme indépendante du matériel utilisé. Le groupe Gpi contient les routines de dessins de PM : lignes, courbes, bitmaps, fontes etc... Les fonctions Win gèrent les fenêtres PM y compris les menus, les barres de défilement et les boîtes de dialogue. Chaque fonction Win est une fonction de « bas niveau », ce qui a souvent pour conséquence une programmation complexe (plusieurs fonctions, chacune avec de nombreux paramètres, doivent être appelées pour effectuer une petite tâche).

En dehors des bibliothèques de fonctions que nous venons de décrire, PM utilise des fichiers-ressources pour stocker des informations spécifiques à l'application concernant les menus, les boites de dialogue, etc. Un fichier ressource est un fichier texte, écrit en code source, qui décrit par exemple un menu déroulant. Les ressources sont compilées (au moyen d'un compilateur de ressources) et si nécessaire reliées à l'application lors de l'édition de liens. Le code source de nombreux types de ressources peut être soit écrit manuellement, soit, par exemple, généré automatiquement par un éditeur des ressources.

L'espace de travai

Vous aurez besoin, pour développer les programmes pour OS/2 et PM, d'un ensemble d'outils assez importants (et coûteux). « L'outil » le plus important sera probablement la documentation. L'ouvrage intitulé « Microsoft OS/2 Programmer's Reference » (volumes 1 à 3) est un ouvrage fondamental. Il existe également un quatrième volume qui traite d'OS/2 version 1.2 - ce volume est resté sur mon étagère parce que j'utilise OS/2 version 1.1. Du fait que les manuels Microsoft sont un peu légers en ce qui concerne les exemples, vous aurez également besoin de vous procurer le livre « Programming the OS/2 Presentation Manager » de Charles Petzold de chez Microsoft Press, ou un livre similaire en français : « Présentation Manager, guide du programmeur » par Gérard Frantz chez Sybex, ainsi que l'un des deux livres suivants sur les noyaux d'OS/2 : « Advanced OS/2 Programming » de Ray Duncan chez Microsoft Press, ou bien « Inside OS/2 » par Peter Norton et Robert Lafore chez Brady Books. « Inside OS/2 » (Microsoft Press, 1988) par Gordon Letwin est également un très bon ouvrage général concernant OS/2. Pour l'utilisation d'OS/2, mais non sa programmation, vous pourriez envisger « Using OS/2 » par Halliday, Minasi, et Gobel (Que Books, 1989).

Vous aurez besoin, bien entendu, d'un compilateur ou d'un assembleur adapté à OS/2 et PM. De nombreux produits Microsoft se trouvent, bien entendu, sur les rangs. On commence à trouver sur le marché quelques autres compilateurs qui supportent OS/2 et PM. Mais quelque soit le langage que vous choisissez d'utiliser, vous devez pouvoir lire du code en langage C - toute la documentation Microsoft est écrite en C.

En ce qui concerne ce projet j'ai utilisé le Modula-2 de Stony Brook, en partie du fait que mon implémentation d'origine du KERMIT ait été écrite en Modula-2 (tout en utilisant le compilateur Logitech), et aussi parce que je préfère le Modula-2 au C. Le compilateur de Stony Brook s'est avéré être un excellent choix - bien qu'une grande partie du système était encore en version bêta-test, je n'ai rencontré qu'un seul problème (que j'ai pu corriger facilement du fait que je disposais du code source du module à corriger).

Vous aurez besoin, enfin, du kit de développement suivant : éditeur d'icônes, éditeur de ressources, éditeur de fontes, compilateur de ressources et de quelques autres outils. Tous ces outils sont disponibles chez Microsoft sous le nom de OS/2 Softset. L'autre solution qui s'offre à vous serait d'acheter le Kit de Développement OS/2 PM qui comporte la documentation Microsoft, l'ouvrage de Petzold, ainsi que les divers outils, mais pas le compilateur. Alors que l'achat séparé

des livres et des outils semble plus économique, le Kit comprend des exemples de programmes ainsi que divers programmes utiles (par exemple PMCAP - un utilitaire de capture d'écran) qui ne sont pas compris avec le Softset.

Format d'un Programme PM

La majorité des GUI, c'est-à-dire GEM, Windows, Macintosh et PM d'0S/2 ont une structure très semblable :

1 - Le programme principal est constitué d'une boucle qui reçoit des messages du système d'exploitation, et les transmet aux procédures de fenêtre.

2 - La procédure de fenêtre trie les différents types de messages, habituellement avec une instruction Case très longue, et les traite en appelant d'autres fonctions.

3 - Toute autre partie du programme qui veut effectuer quelque chose, doit communiquer avec une procédure de fenêtre en « postant » des messages (que le programme principal recherche et réexpédie : voir les étapes 1 et 2).

Tout, pratiquement, est fait au moyen de messages. Un programme PM, par exemple, n'appelle jamais une fonction telle que getchar() ou Read() pour lire un caractère au clavier. Un programme, au contraire, doit rechercher continuellement un message WM_CHAR, et ensuite traduire les paramètres qui sont transmis avec le message, pour comprendre quelle est la touche qui a été frappée.

L'un des énormes avantages d'OS/2 (par rapport aux autres GUI's) est que le multi-tâche préemptif permet à des tâches indépendantes exécutées en arrière plan d'envoyer des messages à la fenêtre principale (pour permettre à l'utilisateur de savoir ce qui se passe). PCKermit utilise des tâches multiples pour l'émulation de terminal (connecté à l'ordinateur maître) ainsi que pour l'envoi et la réception de fichiers.

Mettons-nous au travail !

Le module principal, PCKERMIT.MOD (Listing Un), commence à la ligne 1 (remarquez que les listings sont numérotés de manière séquentielle pour un repérage facile). PCKermit initialise le système de fenêtre et la queue de messages, comporte deux classes de fenêtres (pour la fenêtre principale et une fenêtre enfant - la fenêtre enfant est utilisée pendant les transferts de fichiers), créé et re-dimensionne la fenêtre principale, puis se positionne dans la boucle principale en attendant la fin du programme.

PCKermit est assez traditionnel (pour PM), si ce n'est qu'après avoir créé la fenêtre principale, il détermine immédiatement la dimension et la position du « bureau » (desktop - voir ligne 99), enregistre ceci dans la variable globale Pos, et agrandit la fenêtre à la taille maximale (lignes 100 à 102). La variable Pos est utilisée à nouveau dans le module Shell pour s'assurer que la fenêtre a toujours les dimensions et la position correctes. La fenêtre est soit dimensionnée à trois unités de moins que la taille du « bureau » (pendant le transfert de fichiers) ou agrandie au maximum (pendant l'émulation de terminal). Lorsque la fenêtre est agrandie au maximum, ou qu'elle reprend sa taille normale après avoir été au maximum (en appelant à nouveau la fonction WinSetWindowPos), un terme supplémentaire doit être inclus dans le paramètre final : Soit SWP_RESTORE (lignes 543-544) ou SWP_MAXIMIZE (lignes 552-554). Une stratégie semblable est utilisée pour modifier la taille de la fenêtre enfant (la fenêtre enfant est utilisée pour afficher des messages d'états pendant le transfert de fichiers).

SHELL.DEF (Listing Deux) et SHELL.MOD (Listing Neuf) représentent le module principal puisque c'est là que tous les messages sont traités (principalement dans WindowProc, DoMenu et ChildWindowProc). Plusieurs variables et constantes globales y sont définies (c'est-à-dire Class et ClientWindow). En dehors des deux procédures de fenêtres il existe un certain nombre de procédures annexes pour contrôler les fenêtres : SetFull, SetRestore et SetMaximize contrôlent la dimension de la fenêtre (PCKermit « exige »une fenêtre plein écran pour émuler correctement l'écran vidéo TVI 950); Enable et Disable affichent en grisé tout élément de menu qui n'est pas immédiatement accessible; Check et Uncheck indiquent le mode couleur vidéo actuellement sélectionné; DoMenu est appelé à partir de Window Proc pour traiter les messages WM_COMMAND (ce sont essentiellement des commandes qui résultent de l'interaction entre l'utilisateur et les menus); plusieurs procédures de gestion de boîtes de dialogue (BaudDlgProc, par exemple) permettent à l'utilisateur d'effectuer des choix au moyen de boîte de dialogue pop-up (chaque boîte de dialogue est une fenêtre et nécessite par conséquent sa propre procédure de fenêtre); KeyTranslate traite les messages en provenance du clavier et les traduit en code standard (pour utilisation par le module Term). La procédure de fenêtre WindowProc (ligne 1088) n'est appelée que par PM. Son rôle est de traiter les messages (y compris les messages système standard tels que WM_CREATE, WM_INITMENU, WM_COMMAND et WM_PAINT, ainsi que les messages propres à PCKermit tels que WM_SETFULL, WM_SETRESTORE, et WM_TERM). En dehors du message lui-même (qui n'est en réalité qu'un nombre entier), deux paramètres du message (mp1 et mp2) sont transmis à la procédure de fenêtre. La signification de ces deux paramètres dépend du message lui-même. Par exemple, dans le cas du message WM_TERM, les paramètres du message ne sont utilisés que pour transmettre un seul caractère. Dans le cas du message WM_CHAR, les paramètres contiennent de nombreuses informations sur la touche appuyée (code ASCII, code de balayage, état des touches Control/Alt/Shift, durée de dépression de la touche ainsi que plusieurs drapeaux). Dans certains cas, les paramètres du message ne sont pas utilisés du tout.

L'un des messages traité par WindowProc, WM_COMMAND, est transmis à la fonction DoMenu (ligne 743). Cette procédure traite tous les messages qui résultent d'un « clic » de l'utilisateur sur un menu ou de l'emploi d'un raccourci clavier pour une fonction du menu. Cette interaction avec l'utilisateur a souvent pour conséquence la création d'autres fenêtres enfants, appelées « boîte de dialogue ». Lorsque la boîte de dialogue est affichée à l'écran, une autre procédure de fenêtre prend le relais. Si l'utilisateur clique, par exemple, sur le menu Options, et choisit l'option « vitesse de transmission... », une boîte de dialogue, comportant des boutons radio et un bouton OK, apparaît à l'écran. Lorsque cette boîte de dialogue est à l'écran, la fonction BaudDlgProc, (ligne 863) prend le contrôle de la fenêtre jusqu'à ce que l'utilisateur clique sur OK ou appuie sur l'une des touches Entrée ou Echap. Chacune des boîtes de dialogue ou des boîtes de messages d'alerte (PCKermit en comporte 9) dispose de sa propre procédure de fenêtre.

La procédure de la fenêtre enfant, ChildWindowProc (ligne 1203), n'est également appelée que par PM (tout comme toutes les procédures de fenêtre) et traite un certain nombre de messages provenant du système et de PCKermit. Les messages WM_PAD et PM_DL proviennent respectivement des modules PAD (Packet Assembler Disassembler) et DataLink, et sont utilisés essentiellement pour permettre à ces modules (qui sont exécutés comme des tâches indépendantes) de maintenir l'utilisateur informé de ce qui se passe. Les paramètres du message indiquent quel message doit être affiché à l'écran.

Lorsque l'utilisateur sélectionne Send ou Receive (que ce soit par intermédiaire des menus ou par un raccourci clavier), une suite d'événements relativement complexes intervient. Les messages IDM_SEND ou IDM_REC sont reconnus aux lignes 776 ou 780 de DoMenu. Dans le cas de IDM_SEND, une boîte de dialogue est appelée. La procédure de la boîte de dialogue SendFNDlgProc de la ligne 983 permet à l'utilisateur d'entrer un nom de fichier; PM retourne un pointeur sur le nom de fichier que l'utilisateur a entré.

Les messages IDM_SEND ou IDM_REC provoquent l'appel de la procédure MakeChild (ligne 630). Cette procédure commence par demander à la fenêtre principale d'occuper tout l'écran, puis désactive certaines options du menu (du fait que nous ne voulons pas que l'utilisateur essaie de modifier la vitesse en bauds au milieu d'un transfert de fichiers, ligne 639). Elle crée ensuite une fenêtre standard (ligne 649) dimensionne et positionne la fenêtre (ligne 661), fait apparaître un message adéquat dans la fenêtre (ligne 666) et enfin, rend active la nouvelle fenêtre enfant (ligne 668).

Une fois que la fenêtre est initialisée, nous devons faire en sorte qu'elle puisse envoyer des messages. PCKermit se sert pour cela de AVio, et les lignes 671 à 673 met au point les hvps (Handle to Vio Présentation Space).

De retour dans la fonction DoMenu, une nouvelle tâche est créée aux lignes 779 à 782. OS/2 activera cette tâche (qui enverra ou recevra un fichier) à la prochaine interruption. La tâche se terminera d'elle-même lorsque le transfert du fichier sera terminé, ou quand cinq erreurs consécutives se seront produites. Avant que la tâche ne se termine réellement (en appelant DosExit), elle envoie des messages à sa procédure de fenêtre pour lui demander de fermer la fenêtre (boîte de dialogue) et de restaurer l'état des menus.

Lorsque l'utilisateur sélectionne le mode Connect (que ce soit par intermédiaire des menus ou par un raccourci clavier), le message IDM_CONNECT est reconnu à la ligne 762. Tout comme dans le cas de Send (envoi) et Receive (réception), diverses options des menus sont désactivées et un espace de présentation (hvps) est créé. A la différence de Send et Receive, aucune nouvelle tâche n'est créée, mais une tâche existante est réactivée (ligne 774). Cette tâche a été créée lors de l'initialisation (ligne 1145) puis immédiatement suspendue (ligne suivante).

Prochaine fois

Si vous avez étudié les listings jusqu'à maintenant vous aurez remarqué que je n'ai pas mentionné les listings de trois à huit. Il s'agit des fichiers de définition (.DEF) des modules que nous étudierons dans la deuxième partie de cet article. Le listing trois par exemple, est le fichier .DEF du module Term qui émule le terminal TVI 950. Vous remarquerez également la définition de la procédure Dir (ligne 176) du Listing trois. En plus de l'affichage d'un répertoire, cette fonction permet à l'utilisateur de changer de lecteur de disque ou de passer à un autre répertoire. Cette caractéristique est nécessaire dans un programme Kermit du fait que seul un nom de fichier doit être spécifié lors d'un envoi de fichier (vous ne pouvez pas envoyer un fichier comportant un chemin complet). La seconde partie de cet article étudie ces conséquences en détail. Screen.DEF (Listing quatre) vous donne un avant-goût du module écran qui sera présenté dans la seconde partie. Il effectue des entrées sorties écran de bas niveau au moyen de fonctions VIO de OS/2. Et Comm Port.DEF (Listing sept) présente un module fourni avec le compilateur Stony Brook que j'ai amélioré en ajoutant des tampons supplémentaires pour, par exemple, contourner une limitation OS/2. Bien entendu nous examinerons également quelques problèmes inattendus liés au portage du PCKermit à OS/2. Au mois prochain...

Brian R. Anderson est instructeur au département informatique de l'Institut de Technologie de Colombie Britannique. On peut le joindre à l'adresse suivante : 3700 Willingdon Ave, Burnaby, B.C. British Columbia, Canada V5G 3H2.

KERMIT POUR OS/2 :
copyright **Dr. Dobb's Journal,**
M&T Publishing Inc.
Redwood City,
CA USA 1990.
All right reserved.

Les annonces de O.U.F!

PROCHAINE REUNION : LUNDI 11 MARS 1991.

Cette réunion sera consacrée à "l'autre Windows" : le logiciel ENSEMBLE de Géoworks. Vous avez pu lire dans la presse des annonces consacrées à la version béta américaine 1.0 de cet environnement graphique à moins de $200 qui, à la différence de celui de MicroSoft, peut tourner sur tous les types de compatibles, y compris ceux dotés d'un simple 8088 avec 512 Ko de mémoire et de n'importe quel type d'écran ! C'est Eric Lux, responsable technique de la société "The Disc Company", qui viendra nous présenter la version française de ce logiciel particulièrement convivial.

LE PRIX DES DISQUETTES.

Pour faire face à la concurrence des multiples marchands de disquettes de programmes du domaine public, qui écument le "marché" à grands coups de pages publicitaires, nous avons décidé de resserrer un peu plus notre budget et de diminuer nos prix. Tout d'abord Ouf va s'ouvrir au grand public, en permettant à des non-adhérents d'acquérir des disquettes au détail au prix de Frs 60 TTC. Pour ses membres le prix des disquettes au détail passe à 40 Frs TTC.
Les carnets de 20 disquettes seront délivrés pour un supplément de cotisation de Frs 600,- ce qui ramène le prix de la disquette à Frs 30 l'unité. Le minimum de facturation est maintenu à Frs 120, et les disquettes commandées sur un carnet pourront être expédiées par deux minimum.

Redécouvrez votre informatique

Garantie exclusive

Si, pour une quelconque raison, après que vous ayez reçu l'un de nos logiciels et effectué tous vos tests chez vous pendant toute une semaine entière, celui-ci ne vous procurait pas satisfaction, retournez-le (en bon état, évidemment) et vous serez intégralement remboursé. Ma garantie est formelle.
Satisfait ou totalement remboursé.
Jean-Christophe Krust,Directeur.

⇨ La Productivité

101 Logiciels pour Windows

Une incroyable collection de logiciels graphique/souris. Chacun connait le remarquable environnement graphique/souris choisi par Microsoft comme plateforme du futur. Nous avons réussi l'incroyable tour de force de réunir en un seul coffret 101 programmes d'excellente qualité, pour Windows 2 et 3, qui couvrent tous les domaines. Les meilleurs utilitaires graphiques, de système, de communication. Quelques ludiciels, pour la détente, mais surtout de véritables applications, et des outils dans l'esprit de PC-Tools, sous l'interface Windows. Le tout sélectionné parmi les meilleurs programmes internationaux, et commenté par un grand spécialiste français. 595,00 TTC franco

Gram'R
Plus jamais de fautes

Le traitement de textes d'aujourd'hui détecte parfois les fautes d'orthographe les plus courantes. Gram'R, lui, sait aussi détecter de manière fiable les fautes de grammaire et d'accord. Il offre en plus un conjugueur, et on peut créer soi-même ses propres dictionnaires spécialisés, de taille illimitée. Gram'R relit et corrige directement les formats suivants : ASCII, Word 4 et 5, WordPerfect 5.0 et 5.1, Textor, Wordstar 2000, Microsoft Works. Si l'orthographe et la grammaire ne sont pas votre fort, laissez faire Gram'R ! Edité en France, c'est une remarquable percée technologique. 1399,00 F TTC franco

⇨ Le Système

Lifeguard
Anti-virus, anti-fausses manoeuvres

Ce dispositif logique protège en permanence votre système contre tous lesvirus ou les manoeuvres non-autorisées. Il surveille à tout instant les manoeuvres malveillantes (par exemple les tentatives d'écriture non-autorisées) et les empêche, en vous le signalant immédiatement. Des centaines de sites, dont de nombreuses écoles et universités, sont ainsi déjà protégés contre les virus et les malveillances, passés, présents ou à venir. 590,00 F TTC franco

⇨ Les Loisirs

101 Jeux
Une collection complète en un coffret

La plus importante compilation de l'année. De quoi passer des semaines à découvrir la plus formidable collection de logiciels de jeux proposée en un seul coffret. Depuis les gags les plus géniaux (Bugs, Drain, Virall, Hahal) aux programmes les plus pointus et les plus récents : Spiroïde, Mille Bornes, Alf, Moraff's Pinball, Bananoïd, qui savent faire exploser votre écran de couleurs et de graphiques inouïs ! Il y en a vraiment pour tous les goûts, dans ce véritable must du logiciel ludique. 595 F TTC franco

Généa III
Généalogie informatisée

Permet l'enregistrement dynamique et la manipulation des informations essentielles (plus de 70 rubriques d'information, généalogie ascendante et descendante), les recherches, les listes, l'Arbre génalogique, les formulaires, etc... Présenté auCongrès National de Généalogie (Arras 89). 945,00 F TTC franco

Coquines
Des images...intimes

Interdites aux enfants, ces images couleur d'excellente qualité conviennent à tous les écrans : CGA, EGA, VGA, Hercules, mais pas à touts les yeux, vous voici averti. Bien entendu, les plus fins détails et les couleurs les plus délicates s'expriment au mieux sur les meilleurs écrans. Comme les actrices de ces scènes pour le moins troublantes. A l'heure où certaines télévisions en font tous les samedi soir, il est temps que les PC se mettent au goût du jour. Produit U.S. 180,00 F TTC franco

⇨ Le Minitel

TinyTel et TinyCâble
Des milliers de programmes en libre-service

Emulateur Minitel compatible avec toutes les cartes, modems ou câbles Minitel; toutes les cartes vidéo. TyniTel permet l'accès à tous les serveurs, les mémorisations, impressions de pages et envois de textes. Avec ses utilitaires, accédez en outre à de gigantesques réservoirs de logiciels gratuits en téléchargeant sur l'OI, SM1, PCMAG et VIF (serveurs Minitel en 3615) des programmes pour votre PC : en quelques minutes, les programmes de votre choix. Utilitaires, jeux, outils de dévelo pement. Compatible PC, AT, Windows, PS. Livré "clé en main" avec câble Minitel. 645,00 F TTC franco

MailTel
"Pilleur d'adresses" intelligent

Mailtel transfère sur votre IBM-PC toutes les adresses de l'annuaire électronique (étiquettes et fichiers). Recherche automatique sur toute la France, avec le code postal. Possibilité de mémento téléphonique. La recherche par nom sur la France entière intéressera bien des Notaires et Généalogistes. Livré "clé en main", avec câble Minitel. 945,00 F TTC franco

Cadeau !

Avec votre commande, en cadeau gratuit est définitif, vous recevrez PKLITE, un étonnant logiciel qui comprime les programmes, jusqu'à 40% de leur taille originale, en les gardant exécutables. Idéal pour faire tenir un programme sur une disquette. En outre, le code source est protégé, impossible à désassembler ou à modifier, une garantie pour le programmeur.

✂ **BON DE COMMANDE** - Catalogue gratuit sur demande. *Distribution par correspondance, ou directement dans les boutiques micro et les Fnac.*
Oui ! Je commande immédiatement les produits suivants, qui me seront livrés par retour du courrier, en expédition Poste Colissimo. Je recevrai également mon cadeau exceptionnel : PKLITE, le compacteur de logiciels exécutables.

Qté Nom du produit Prix TTC

... .

Société/Nom .....
Prénom .....
Adresse .....
Code Postal ..... Ville .....
Tél .. ..

Règlement par [] Chèque ci-joint
[] Carte Bleue/Visa N°.....
Date de validité ../..
SIGNATURE OBLIGATOIRE
pour carte bancaire [_____]

Commandes par téléphone : (1)40 04 90 34
(1)40 04 90 31
(Province) 86 74 32 23

Télécopie : (1)40 04 90 32

LOGICIELS ET MÉDIAS - 125, rue de Saussure - 75017 PARIS

[SERVICE LECTEURS : INTERESSANT = 7/ MOYEN = 8 / MAUVAIS =9]

DEBUG magazine

CAHIER

LISTINGS

SOMMAIRE

10.1 : Kermit pour O/S 2 p 2

10.2 : Protect p 13

10.3 : Un Analyseur de Protocole Transmission Série p 16
(Article paru dans le n° 9 du mois de Janvier 1991).

10.1 : KERMIT pour OS/2

KERMIT pour OS/2
par Brian R. Anderson

[LISTING UN]

```
MODULE PCKermit;

(*************************************************************************)
(*                                                                       *)
(*                  PCKermit  -  par Brian R. Anderson                   *)
(*                        Copyright (c) 1990                             *)
(*                                                                       *)
(*  PCKermit est l'implémentation du protocole de transfert de fichiers  *)
(*  developpé à Columbia University.  cette version (OS/2 PM) est le     *)
(*  portage de la version DOS, que j'ai écrite précédemment.             *)
(*                                                                       *)
(*  La présente version inclut l'émulation du terminal TVI 950, pour     *)
(*  un utilisation avec les mainframes IBM (jusqu'à IBM 7171).           *)
(*                                                                       *)
(*************************************************************************)

   FROM SYSTEM IMPORT
      ADR;

   FROM OS2DEF IMPORT
      HAB, HWND, HPS, NULL, ULONG;

   FROM PMWIN IMPORT
      MPFROM2SHORT, HMQ, QMSG, CS_SIZEREDRAW,  WS_VISIBLE, FS_ICON,
      FCF_TITLEBAR, FCF_SYSMENU, FCF_SIZEBORDER, FCF_MINMAX, FCF_ACCELTABLE,
      FCF_SHELLPOSITION, FCF_TASKLIST, FCF_MENU, FCF_ICON,
      SWP_MOVE, SWP_SIZE, SWP_MAXIMIZE,
      HWND_DESKTOP, FID_SYSMENU, SC_CLOSE, MIA_DISABLED, MM_SETITEMATTR,
      WinInitialize, WinCreateMsgQueue, WinGetMsg, WinDispatchMsg, WinSendMsg,
      WinRegisterClass, WinCreateStdWindow, WinDestroyWindow, WinWindowFromID,
      WinDestroyMsgQueue, WinTerminate, WinSetWindowText,
      WinSetWindowPos, WinQueryWindowPos;

   FROM KH IMPORT
      IDM_KERMIT;

   FROM Shell IMPORT
      Class, Title, Child, WindowProc, ChildWindowProc,
      FrameWindow, ClientWindow, SetPort, Pos;


   CONST
      QUEUE_SIZE = 1024;    (* Grande queue de messages pour
                               évènements asynchrones *)

   VAR
      AnchorBlock : HAB;
      MessageQueue : HMQ;
      Message : QMSG;
      FrameFlags : ULONG;
      hsys : HWND;


BEGIN    (* main - routine principale *)
   AnchorBlock := WinInitialize(0);

   IF AnchorBlock # 0 THEN
      MessageQueue := WinCreateMsgQueue (AnchorBlock, QUEUE_SIZE);

      IF MessageQueue # 0 THEN
         (* Enregistre la classe de fenêtre parent *)
         WinRegisterClass (
             AnchorBlock,
             ADR (Class),
             WindowProc,
             CS_SIZEREDRAW, 0);

         (* Enregistre la classe de fenêtre enfant *)
         WinRegisterClass (
             AnchorBlock,
             ADR (Child),
             ChildWindowProc,
             CS_SIZEREDRAW, 0);

         (* Crée une fenêtre standard *)
         FrameFlags := FCF_TITLEBAR + FCF_MENU + FCF_MINMAX +
                       FCF_SYSMENU + FCF_SIZEBORDER + FCF_TASKLIST +
                       FCF_ICON + FCF_SHELLPOSITION + FCF_ACCELTABLE;

         FrameWindow := WinCreateStdWindow (
                  HWND_DESKTOP,           (* handle de la fenêtre parent *)
                  WS_VISIBLE + FS_ICON,   (* style de la fenêtre *)
                  FrameFlags,             (* attributs *)
                  ADR(Class),             (* classe de la fenêtre *)
                  NULL,                   (* titre de la fenêtre *)
                  WS_VISIBLE,             (* style de la fenetre client *)
                  NULL,                   (* handle du module ressources *)
                  IDM_KERMIT,             (* id des ressources *)
                  ClientWindow            (* handle de fenêtre client
                                             retourné par la fonction *)
         IF FrameWindow # 0 THEN
            (* Interdit l'usage de l'icône de fermeture dans menu systeme *)
            hsys := WinWindowFromID (FrameWindow, FID_SYSMENU);
            WinSendMsg (hsys, MM_SETITEMATTR,
               MPFROM2SHORT (SC_CLOSE, 1),
               MPFROM2SHORT (MIA_DISABLED, MIA_DISABLED));

            (* Maximise la taille de la fenêtre et affiche le titre *)
            WinQueryWindowPos (HWND_DESKTOP, ADR (Pos));
            WinSetWindowPos (FrameWindow, 0,
               Pos.x + 3, Pos.y + 3, Pos.cx - 6, Pos.cy - 6,
               SWP_MOVE + SWP_SIZE);
            WinSetWindowText (FrameWindow, ADR (Title));

            SetPort;   (* Tente l'initialisation du port de communications *)

            WHILE WinGetMsg(AnchorBlock, Message, NULL, 0, 0) # 0 DO
               WinDispatchMsg(AnchorBlock, Message);
            END;

            WinDestroyWindow(FrameWindow);
         END;
         WinDestroyMsgQueue(MessageQueue);
      END;
      WinTerminate(AnchorBlock);
   END;
END PCKermit.
```

[LISTING DEUX]

```
DEFINITION MODULE Shell;

   FROM OS2DEF IMPORT
      USHORT, HWND;

   FROM PMWIN IMPORT
      MPARAM, MRESULT, SWP;

   EXPORT QUALIFIED
      Class, Child, Title, FrameWindow, ClientWindow,
      ChildFrameWindow, ChildClientWindow, Pos, SetPort,
      WindowProc, ChildWindowProc;

   CONST
      Class = "PCKermit";
      Child ="Child";
      Title = "PCKermit - Communication entre Micros et Mainframes";


   VAR
      FrameWindow : HWND;
      ClientWindow : HWND;
      ChildFrameWindow : HWND;
      ChildClientWindow : HWND;
      Pos : SWP;    (* Dimensions de l'écran : position & taille *)
      comport : CARDINAL;


   PROCEDURE SetPort;

   PROCEDURE WindowProc ['WindowProc'] (
      hwnd : HWND;
      msg  : USHORT;
      mp1  : MPARAM;
      mp2  : MPARAM) : MRESULT [LONG, LOADDS];

   PROCEDURE ChildWindowProc ['ChildWindowProc'] (
      hwnd : HWND;
      msg  : USHORT;
      mp1  : MPARAM;
      mp2  : MPARAM) : MRESULT [LONG, LOADDS];

END Shell.
```

[LISTING TROIS]

```
DEFINITION MODULE Term;    (* Emulation de terminal TVI950 pour Kermit *)

   EXPORT QUALIFIED
      WM_TERM, WM_TERMQUIT,
      Dir, TermThrProc, InitTerm, PutKbdChar, PutPortChar;

   CONST
      WM_TERM = 4000H;
      WM_TERMQUIT = 4001H;


   PROCEDURE Dir (path : ARRAY OF CHAR);
   (* Affiche un répertoire *)

   PROCEDURE TermThrProc;
   (* Lit les caractères sur le port série, les range dans un buffer, *)
   (* et envoie un message *)

   PROCEDURE InitTerm;
   (* Efface l'écran, prépare pour l'émulation du terminal *)
```

```
   PROCEDURE PutKbdChar (ch1, ch2 : CHAR);
   (* Traite un caractère lu au clavier *)

   PROCEDURE PutPortChar (ch : CHAR);
   (* Traite un caractère lu sur le port série *)

END Term.
```

[LISTING QUATRE]

```
DEFINITION MODULE Screen;
(* Fonctions d'affichage écran de "bas niveau" utilisant l'AVIO *)

   FROM PMAVIO IMPORT
      HVPS;

   EXPORT QUALIFIED
      NORMAL, HIGHLIGHT, REVERSE, attribute, ColorSet, hvps,
      White, Green, Amber, Color1, Color2,
      ClrScr, ClrEol, GotoXY, GetXY,
      Right, Left, Up, Down, Write, WriteLn, WriteString,
      WriteInt, WriteHex, WriteAtt;


   VAR
      NORMAL : CARDINAL;
      HIGHLIGHT : CARDINAL;
      REVERSE : CARDINAL;
      attribute : CARDINAL;
      ColorSet : CARDINAL;
      hvps : HVPS;    (* espace de présentation utilisé par module screen *)


   PROCEDURE White;
   (* Initialise couleurs : blanc sur noir *)

   PROCEDURE Green;
   (* Initialise couleurs : vert sur noir *)

   PROCEDURE Amber;
   (* Initialise couleurs : ambre sur noir *)

   PROCEDURE Color1;
   (* Initialise couleurs : Bleu, Rouge, Vert *)

   PROCEDURE Color2;
   (* Initialise couleurs : Vert, Magenta, Cyan *)

   PROCEDURE ClrScr;
   (* Efface l'écran *)

   PROCEDURE ClrEol;
   (* Efface la fin de la ligne courante *)

   PROCEDURE Right;
   (* Déplace le curseur vers la droite *)

   PROCEDURE Left;
   (* Déplace le curseur vers la gauche *)

   PROCEDURE Up;
   (* Déplace le curseur vers le haut *)

   PROCEDURE Down;
   (* Déplace le curseur vers le bas *)

   PROCEDURE GotoXY (col, row : CARDINAL);
   (* positionne le curseur *)

   PROCEDURE GetXY (VAR col, row : CARDINAL);
   (* Détermine la position de curseur *)

   PROCEDURE Write (c : CHAR);
   (* Ecrit un caractère, mode télétype *)

   PROCEDURE WriteString (str : ARRAY OF CHAR);
   (* Ecrit une chaine de caractères, mode télétype *)

   PROCEDURE WriteInt (n : INTEGER; s : CARDINAL);
   (* Ecrit un nombre entier, mode télétype *)

   PROCEDURE WriteHex (n, s : CARDINAL);
   (* Ecrit un nombre hexa, mode télétype *)

   PROCEDURE WriteLn;
   (* Ecrit <cr> <lf>, mode télétype *)

   PROCEDURE WriteAtt (c : CHAR);
   (* Ecrit un caractère et ses attributs à la position du curseur *)

END Screen.
```

[LISTING CINQ]

```
DEFINITION MODULE PAD;    (* Assembleur/Désassembleur de paquets *)

   FROM PMWIN IMPORT
      MPARAM;

   EXPORT QUALIFIED
      WM_PAD, PAD_Quit, PAD_Error, PacketType, yourNPAD, yourPADC, yourEOL,
      Aborted, sFname, Send, Receive, DoPADMsg;

   CONST
      WM_PAD = 5000H;
      PAD_Quit = 0;
      PAD_Error = 20;

   TYPE
      (* PacketType est utilisé dans les modules PAD et DataLink *)
      PacketType = ARRAY [1..100] OF CHAR;

   VAR
      (* yourNPAD, yourPADC, et yourEOL utilisés dans PAD et DataLink *)
      yourNPAD : CARDINAL;    (* nombre de caractères de remplissage *)
      yourPADC : CHAR;        (* caractères de remplissage *)
      yourEOL  : CHAR;        (* End Of Line — fin de ligne *)
      sFname : ARRAY [0..20] OF CHAR;
      Aborted : BOOLEAN;

   PROCEDURE Send;
   (* Envoie un fichier après avoir demandé son nom *)

   PROCEDURE Receive;
   (* Reçoit un ou plusieurs fichiers *)

   PROCEDURE DoPADMsg (mp1, mp2 : MPARAM);
   (* Affiche les messages de l'Assembleur/Désassembleur de paquets *)

END PAD.
```

[LISTING SIX]

```
DEFINITION MODULE DataLink;    (* Envoie et reçoit les paquets *)

   FROM PMWIN IMPORT
      MPARAM;

   FROM PAD IMPORT
      PacketType;

   EXPORT QUALIFIED
      WM_DL, FlushUART, SendPacket, ReceivePacket, DoDLMsg;

   CONST
      WM_DL = 6000H;

   PROCEDURE FlushUART;
   (* Vide les caractères qui restent dans les registres "hold" de l'uart *)

   PROCEDURE SendPacket (s : PacketType);
   (* Ajoute SOH et CheckSum au paquet *)

   PROCEDURE ReceivePacket (VAR r : PacketType) : BOOLEAN;
   (* supprime SOH et checksum — retourne un status: TRUE= paquet correct *)
   (* FALSE = time out (paquet non reçu) ou erreur de checksum *)

   PROCEDURE DoDLMsg (mp1, mp2 : MPARAM);
   (* Traite les messages du DataLink *)

END DataLink.
```

[LISTING SEPT]

```
(*************************************************************)
(*                                                           *)
(*                Copyright (C) 1988, 1989                   *)
(*                  Stony Brook Software                     *)
(*                                                           *)
(*                   Tous droits réservés                    *)
(*                                                           *)
(*************************************************************)

DEFINITION MODULE CommPort;

   TYPE
      CommStatus = (
               Success,
               InvalidPort,
               InvalidParameter,
               AlreadyReceiving,
               NotReceiving,
               NoCharacter,
               FramingError,
               OverrunError,
               ParityError,
               BufferOverflow,
               TimeOut
      );

      BaudRate = (
               Baud110,
               Baud150,
               Baud300,
               Baud600,
               Baud1200,
               Baud2400,
               Baud4800,
               Baud9600,
               Baud19200
      );

      DataBits = [7..8];
      StopBits = [1..2];
      Parity = (Even, Odd, None);
```

```
   PROCEDURE InitPort(port : CARDINAL; speed : BaudRate; data : DataBits;
                      stop : StopBits; check : Parity) : CommStatus;

   PROCEDURE StartReceiving(port, bufsize : CARDINAL) : CommStatus;

   PROCEDURE StopReceiving(port : CARDINAL) : CommStatus;

   PROCEDURE GetChar(port : CARDINAL; VAR ch : CHAR) : CommStatus;

   PROCEDURE SendChar(port : CARDINAL; ch : CHAR;
                      modem : BOOLEAN) : CommStatus;

END CommPort.
```

[LISTING HUIT]

```
DEFINITION MODULE Files;   (* Gestion E/S fichiers pour Kermit *)

   FROM FileSystem IMPORT
      File;

   EXPORT QUALIFIED
      Status, FileType, Open, Create, CloseFile, Get, Put, DoWrite;

   TYPE
      Status = (Done, Error, EOF);
      FileType = (Input, Output);

   PROCEDURE Open (VAR f : File; name : ARRAY OF CHAR) : Status;
   (* ouvre un fichier existant en lecture, retourne un status *)

   PROCEDURE Create (VAR f : File; name : ARRAY OF CHAR) : Status;
   (* crée un nouveau fichier en écriture, retourne un status *)

   PROCEDURE CloseFile (VAR f : File; Which : FileType) : Status;
   (* Ferme un fichier après lecture ou écriture *)

   PROCEDURE Get (VAR f : File; VAR ch : CHAR) : Status;
   (* Lit un caractère dans un fichier, retourne un status *)

   PROCEDURE Put (ch : CHAR);
   (* Ecrit un caractère dans le buffer fichier *)

   PROCEDURE DoWrite (VAR f : File) : Status;
   (* Ecrit le buffer fichier sur disque s'il est presque plein *)

END Files.
```

[LISTING NEUF]

```
IMPLEMENTATION MODULE Shell;

   FROM SYSTEM IMPORT
      ADDRESS, ADR;

   IMPORT ASCII;

   FROM OS2DEF IMPORT
      LOWORD, HIWORD, HWND, HDC, HPS, RECTL, USHORT, NULL, ULONG;

   FROM Term IMPORT
      WM_TERM, WM_TERMQUIT,
      Dir, TermThrProc, InitTerm, PutKbdChar, PutPortChar;

   FROM PAD IMPORT
      WM_PAD, PAD_Quit, PAD_Error, DoPADMsg, Aborted, sFname, Send, Receive;

   FROM DataLink IMPORT
      WM_DL, DoDLMsg;

   FROM Screen IMPORT
      hvps, ColorSet, White, Green, Amber, Color1, Color2, ClrScr, WriteLn;

   FROM DosCalls IMPORT
      DosCreateThread, DosSuspendThread, DosResumeThread, DosSleep;

   FROM PMAVIO IMPORT
    VioCreatePS, VioAssociate, VioDestroyPS, VioShowPS, WinDefAVioWindowProc,
    FORMAT_CGA, HVPS;

   FROM PMWIN IMPORT
      MPARAM, MRESULT, SWP, PSWP,
      WS_VISIBLE, FCF_TITLEBAR, FCF_SIZEBORDER, FCF_SHELLPOSITION,
      WM_SYSCOMMAND, WM_MINMAXFRAME, SWP_MINIMIZE, HWND_DESKTOP,
      WM_PAINT, WM_QUIT, WM_COMMAND, WM_INITDLG, WM_CONTROL, WM_HELP,
      WM_INITMENU, WM_SIZE, WM_DESTROY, WM_CREATE, WM_CHAR,
      BM_SETCHECK, MBID_OK, MB_OK, MB_OKCANCEL,
      KC_CHAR, KC_CTRL, KC_VIRTUALKEY, KC_KEYUP,
      SWP_SIZE, SWP_MOVE, SWP_MAXIMIZE, SWP_RESTORE,
      MB_ICONQUESTION, MB_ICONASTERISK, MB_ICONEXCLAMATION,
      FID_MENU, MM_SETITEMATTR, MM_QUERYITEMATTR,
      MIA_DISABLED, MIA_CHECKED, MPFROM2SHORT,
      WinCreateStdWindow, WinDestroyWindow,
      WinOpenWindowDC, WinSendMsg, WinQueryDlgItemText, WinInvalidateRect,
      WinDefWindowProc, WinBeginPaint, WinEndPaint, WinQueryWindowRect,
      WinSetWindowText, WinSetFocus, WinDlgBox, WinDefDlgProc, WinDismissDlg,
      WinMessageBox, WinPostMsg, WinWindowFromID, WinSendDlgItemMsg,
      WinSetWindowPos, WinSetActiveWindow;

   FROM PMGPI IMPORT
      GpiErase;

FROM KH IMPORT
    IDM_KERMIT, IDM_FILE, IDM_OPTIONS, IDM_SENDFN, ID_SENDFN,
    IDM_DIR, IDM_CONNECT, IDM_SEND, IDM_REC, IDM_DIRPATH, ID_DIRPATH,
    IDM_DIREND, IDM_QUIT, IDM_ABOUT, IDM_HELPMENU, IDM_TERMHELP,
    IDM_COMPORT, IDM_BAUDRATE, IDM_DATABITS, IDM_STOPBITS, IDM_PARITY,
    COM_OFF, ID_COM1, ID_COM2, PARITY_OFF, ID_EVEN, ID_ODD, ID_NONE,
    DATA_OFF, ID_DATA7, ID_DATA8, STOP_OFF, ID_STOP1, ID_STOP2,
    BAUD_OFF, ID_B110, ID_B150, ID_B300, ID_B600, ID_B1200, ID_B2400,
    ID_B4800, ID_B9600, ID_B19K2,
    IDM_COLORS, IDM_WHITE, IDM_GREEN, IDM_AMBER, IDM_C1, IDM_C2;

FROM CommPort IMPORT
    CommStatus, BaudRate, DataBits, StopBits, Parity, InitPort,
    StartReceiving, StopReceiving;

FROM Strings IMPORT
    Assign, Append, AppendChar;


CONST
    WM_SETMAX = 7000H;
    WM_SETFULL = 7001H;
    WM_SETRESTORE = 7002H;
    NONE = 0;    (* aucun port n'est encore initialisé *)
    STKSIZE = 4096;
    BUFSIZE = 4096;  (* Buffers de réception : de la place pour
                        deux écrans entiers *)
    PortError = "Port déjà utilisé — FIN? (ESC = essaier un autre port)";
    ESC = 33C;


VAR
    FrameFlags : ULONG;
    TermStack : ARRAY [1..STKSIZE] OF CHAR;
    Stack : ARRAY [1..STKSIZE] OF CHAR;
    TermThr : CARDINAL;
    Thr : CARDINAL;
    hdc : HDC;
    frame_hvps, child_hvps : HVPS;
    TermMode : BOOLEAN;
    Path : ARRAY [0..60] OF CHAR;
    Banner : ARRAY [0..40] OF CHAR;
    PrevComPort : CARDINAL;
    Settings : ARRAY [0..1] OF RECORD
                                  baudrate : CARDINAL;
                                  databits : CARDINAL;
                                  parity : CARDINAL;
                                  stopbits : CARDINAL;
                               END;

PROCEDURE SetFull;
(* Fenêtre pleine taille *)
   BEGIN
      WinSetWindowPos (FrameWindow, 0,
         Pos.x + 3, Pos.y + 3, Pos.cx - 6, Pos.cy - 6,
         SWP_MOVE + SWP_SIZE);
   END SetFull;


PROCEDURE SetRestore;
(* Fenêtre pleine taille après avoir été maximisée *)
   BEGIN
      WinSetWindowPos (FrameWindow, 0,
         Pos.x + 3, Pos.y + 3, Pos.cx - 6, Pos.cy - 6,
         SWP_MOVE + SWP_SIZE + SWP_RESTORE);
   END SetRestore;


PROCEDURE SetMax;
(* Maximise une fenêtre *)
   BEGIN
      WinSetWindowPos (FrameWindow, 0,
         Pos.x + 3, Pos.y + 3, Pos.cx - 6, Pos.cy - 6,
         SWP_MOVE + SWP_SIZE + SWP_MAXIMIZE);
   END SetMax;


PROCEDURE SetBanner;
(* Affiche nom du programme abrégé plus numéro du port com *)
(* dans la barre de titre *)

   CONST
      PortName : ARRAY [0..1] OF ARRAY [0..5] OF CHAR =
         [["COM1:", 0C], ["COM2:", 0C]];
      BaudName : ARRAY [0..8] OF ARRAY [0..5] OF CHAR =
         [["110", 0C], ["150", 0C], ["300", 0C],
          ["600", 0C], ["1200", 0C], ["2400", 0C],
          ["4800", 0C], ["9600", 0C], ["19200", 0C]];
      ParityName : ARRAY [0..2] OF CHAR = ['E', 'O', 'N'];

   BEGIN
      WITH Settings[comport - COM_OFF] DO
         Assign (Class, Banner);
         Append (Banner, " — ");
         Append (Banner, PortName[comport - COM_OFF]);
         Append (Banner, BaudName[baudrate - BAUD_OFF]);
         AppendChar (Banner, ',');
         AppendChar (Banner, ParityName[parity - PARITY_OFF]);
         AppendChar (Banner, ',');
         AppendChar (Banner, CHR ((databits - DATA_OFF) + 30H));
         AppendChar (Banner, ',');
         AppendChar (Banner, CHR ((stopbits - STOP_OFF) + 30H));
         WinSetWindowText (FrameWindow, ADR (Banner));
      END;
   END SetBanner;
```

```
PROCEDURE SetPort;
(* Initialise le port de communication *)
(* avec les paramètres choisis par l'utilisateur *)

   VAR
      status : CommStatus;
      rc : USHORT;

   BEGIN
      IF PrevComPort # NONE THEN
         StopReceiving (PrevComPort - COM_OFF);
      END;

      WITH Settings[comport - COM_OFF] DO
         status := InitPort (
            comport - COM_OFF,
            BaudRate (baudrate - BAUD_OFF),
            DataBits (databits - DATA_OFF),
            StopBits (stopbits - STOP_OFF),
            Parity (parity - PARITY_OFF),
         );
      END;

      IF status = Success THEN
         StartReceiving (comport - COM_OFF, BUFSIZE);
         PrevComPort := comport;
      ELSE
         rc := WinMessageBox (HWND_DESKTOP, FrameWindow, ADR (PortError),
                              0, 0, MB_OKCANCEL + MB_ICONEXCLAMATION);
         IF rc = MBID_OK THEN
            WinPostMsg (FrameWindow, WM_QUIT, 0, 0);
         ELSE   (* essaie l'autre port *)
            IF comport = ID_COM1 THEN
               comport := ID_COM2;
            ELSE
               comport := ID_COM1;
            END;
            SetPort;   (* appel récursif pour réessayer *)
         END;
      END;
      SetBanner;
   END SetPort;


PROCEDURE MakeChild (msg : ARRAY OF CHAR);
(* Crée une fenêtre enfant utilisée par les tâches d'émission *)
(* et de réception *)

   VAR
      c_hdc : HDC;

   BEGIN
      WinPostMsg (FrameWindow, WM_SETFULL, 0, 0);

      Disable (IDM_CONNECT);
      Disable (IDM_SEND);
      Disable (IDM_REC);
      Disable (IDM_DIR);
      Disable (IDM_OPTIONS);
      Disable (IDM_COLORS);

      (* Crée la fenêtre client *)
      FrameFlags := FCF_TITLEBAR + FCF_SIZEBORDER;

      ChildFrameWindow := WinCreateStdWindow (
               ClientWindow,        (* handle de la fenêtre parent *)
               WS_VISIBLE,          (* style de la fenêtre *)
               FrameFlags,          (* attributs *)
               ADR(Child),          (* classe de la fenêtre *)
               NULL,                (* titre de la fenêtre *)
               WS_VISIBLE,          (* style de la fenetre client *)
               NULL,                (* handle du module des ressources *)
               IDM_KERMIT,          (* id des ressources *)
               ChildClientWindow    (* handle de fenêtre client
                                       retourné par la fonction *)
      );

      WinSetWindowPos (ChildFrameWindow, 0,
         Pos.cx DIV 4, Pos.cy DIV 4,
         Pos.cx DIV 2, Pos.cy DIV 2 - 3,
         SWP_MOVE + SWP_SIZE);

      WinSetWindowText (ChildFrameWindow, ADR (msg));

      WinSetActiveWindow (HWND_DESKTOP, ChildFrameWindow);

      c_hdc := WinOpenWindowDC (ChildClientWindow);
      hvps := child_hvps;
      VioAssociate (c_hdc, hvps);
      ClrScr;      (* efface la fenêtre hvio *)
   END MakeChild;


PROCEDURE Disable (item : USHORT);
(* inhibe et affiche un menu en gris *)

   VAR
      h : HWND;

   BEGIN
      h := WinWindowFromID (FrameWindow, FID_MENU);
      WinSendMsg (h, MM_SETITEMATTR,
         MPFROM2SHORT (item, 1),
         MPFROM2SHORT (MIA_DISABLED, MIA_DISABLED));
   END Disable;


PROCEDURE Enable (item : USHORT);
(* autorise une entrée de menu *)

   VAR
      h : HWND;
      atr : USHORT;

   BEGIN
      h := WinWindowFromID (FrameWindow, FID_MENU);
      atr := USHORT (WinSendMsg (h, MM_QUERYITEMATTR,
                     MPFROM2SHORT (item, 1),
                     MPFROM2SHORT (MIA_DISABLED, MIA_DISABLED)));
      atr := USHORT (BITSET (atr) * (BITSET (MIA_DISABLED) / BITSET (-1)));
      WinSendMsg (h, MM_SETITEMATTR,
         MPFROM2SHORT (item, 1),
         MPFROM2SHORT (MIA_DISABLED, atr));
   END Enable;


PROCEDURE Check (item : USHORT);
(* Affiche une marque devant un menu *)
(* Indique qu'il est marqué *)

   VAR
      h : HWND;

   BEGIN
      h := WinWindowFromID (FrameWindow, FID_MENU);
      WinSendMsg (h, MM_SETITEMATTR,
         MPFROM2SHORT (item, 1),
         MPFROM2SHORT (MIA_CHECKED, MIA_CHECKED));
   END Check;


PROCEDURE UnCheck (item : USHORT);
(* Supprime une marque devant un menu *)

   VAR
      h : HWND;
      atr : USHORT;

   BEGIN
      h := WinWindowFromID (FrameWindow, FID_MENU);
      atr := USHORT (WinSendMsg (h, MM_QUERYITEMATTR,
                     MPFROM2SHORT (item, 1),
                     MPFROM2SHORT (MIA_CHECKED, MIA_CHECKED)));
      atr := USHORT (BITSET (atr) * (BITSET (MIA_CHECKED) / BITSET (-1)));
      WinSendMsg (h, MM_SETITEMATTR,
         MPFROM2SHORT (item, 1),
         MPFROM2SHORT (MIA_CHECKED, atr));
   END UnCheck;


PROCEDURE DoMenu (hwnd : HWND; item : MPARAM);
(* Traite les interactions menus *)

   VAR
      rcl : RECTL;
      rc : USHORT;

   BEGIN
      CASE  LOWORD (item) OF
         IDM_DIR:
           SetFull;
           WinQueryWindowRect (hwnd, rcl);
           WinDlgBox (HWND_DESKTOP, hwnd, PathDlgProc, 0, IDM_DIRPATH, 0);
           hvps := frame_hvps;
           VioAssociate (hdc, hvps);
           Dir (Path);
           WinDlgBox (HWND_DESKTOP, hwnd, DirEndDlgProc, 0, IDM_DIREND, 0);
           VioAssociate (0, hvps);
           WinInvalidateRect (hwnd, ADR (rcl), 0);
       | IDM_CONNECT:
           TermMode := TRUE;
           Disable (IDM_CONNECT);
           Disable (IDM_SEND);
           Disable (IDM_REC);
           Disable (IDM_DIR);
           Disable (IDM_OPTIONS);
           Disable (IDM_COLORS);
           (* MAXIMIZE une fenêtre — Nécessaire pour émulation Terminal *)
           SetMax;
           hvps := frame_hvps;
           VioAssociate (hdc, hvps);
           DosResumeThread (TermThr);
           InitTerm;
       | IDM_SEND:
          WinDlgBox (HWND_DESKTOP, hwnd, SendFNDlgProc, 0, IDM_SENDFN, 0);
          MakeChild ("Envoie un fichier");
          DosCreateThread (Send, Thr, ADR (Stack[STKSIZE]));
       | IDM_REC:
           MakeChild ("Reçoit un fichier");
           DosCreateThread (Receive, Thr, ADR (Stack[STKSIZE]));
       | IDM_QUIT:
           rc := WinMessageBox (HWND_DESKTOP, ClientWindow,
              ADR ("Voulez vous quitter PCKermit?"),
              ADR ("Fin de session"), 0, MB_OKCANCEL + MB_ICONQUESTION);
           IF rc = MBID_OK THEN
              StopReceiving (comport - COM_OFF);
              WinPostMsg (hwnd, WM_QUIT, 0, 0);
           END;
       | IDM_COMPORT:
```

```
            WinDlgBox (HWND_DESKTOP, hwnd, ComDlgProc, 0, IDM_COMPORT, 0);
            SetPort;
      |  IDM_BAUDRATE:
            WinDlgBox (HWND_DESKTOP, hwnd, BaudDlgProc, 0, IDM_BAUDRATE, 0);
            SetPort;
      |  IDM_DATABITS:
            WinDlgBox (HWND_DESKTOP, hwnd, DataDlgProc, 0, IDM_DATABITS, 0);
            SetPort;
      |  IDM_STOPBITS:
            WinDlgBox (HWND_DESKTOP, hwnd, StopDlgProc, 0, IDM_STOPBITS, 0);
            SetPort;
      |  IDM_PARITY:
            WinDlgBox (HWND_DESKTOP, hwnd, ParityDlgProc, 0, IDM_PARITY, 0);
            SetPort;
      |  IDM_WHITE:
            UnCheck (ColorSet);
            ColorSet := IDM_WHITE;
            Check (ColorSet);
            White;
      |  IDM_GREEN:
            UnCheck (ColorSet);
            ColorSet := IDM_GREEN;
            Check (ColorSet);
            Green;
      |  IDM_AMBER:
            UnCheck (ColorSet);
            ColorSet := IDM_AMBER;
            Check (ColorSet);
            Amber;
      |  IDM_C1:
            UnCheck (ColorSet);
            ColorSet := IDM_C1;
            Check (ColorSet);
            Color1;
      |  IDM_C2:
            UnCheck (ColorSet);
            ColorSet := IDM_C2;
            Check (ColorSet);
            Color2;
      |  IDM_ABOUT:
            WinDlgBox (HWND_DESKTOP, hwnd, AboutDlgProc, 0, IDM_ABOUT, 0);
      ELSE
         (* Ne fait rien... *)
      END;
   END DoMenu;


PROCEDURE ComDlgProc ['ComDlgProc'] (
(* Gère une boîte de dialogue pour choisir COM1/COM2 *)
      hwnd  : HWND;
      msg   : USHORT;
      mp1   : MPARAM;
      mp2   : MPARAM) : MRESULT [LONG, LOADDS];
   BEGIN
      CASE msg OF
         WM_INITDLG:
            WinSendDlgItemMsg (hwnd, comport, BM_SETCHECK, 1, 0);
            WinSetFocus (HWND_DESKTOP, WinWindowFromID (hwnd, comport));
            RETURN 1;
      |  WM_CONTROL:
            comport := LOWORD (mp1);
            RETURN 0;
      |  WM_COMMAND:
            WinDismissDlg (hwnd, 1);
            RETURN 0;
      ELSE
         RETURN WinDefDlgProc (hwnd, msg, mp1, mp2);
      END;
   END ComDlgProc;


PROCEDURE BaudDlgProc ['BaudDlgProc'] (
(* Gère une boîte de dialogue pour choisir la vitesse en bauds *)
      hwnd  : HWND;
      msg   : USHORT;
      mp1   : MPARAM;
      mp2   : MPARAM) : MRESULT [LONG, LOADDS];
   BEGIN
      WITH Settings[comport - COM_OFF] DO
         CASE msg OF
            WM_INITDLG:
               WinSendDlgItemMsg (hwnd, baudrate, BM_SETCHECK, 1, 0);
               WinSetFocus (HWND_DESKTOP, WinWindowFromID (hwnd, baudrate));
               RETURN 1;
         |  WM_CONTROL:
               baudrate := LOWORD (mp1);
               RETURN 0;
         |  WM_COMMAND:
               WinDismissDlg (hwnd, 1);
               RETURN 0;
         ELSE
            RETURN WinDefDlgProc (hwnd, msg, mp1, mp2);
         END;
      END;
   END BaudDlgProc;


PROCEDURE DataDlgProc ['DataDlgProc'] (
(* Gère une boîte de dialogue pour choisir 7 ou 8 bits de données *)
      hwnd  : HWND;
      msg   : USHORT;
      mp1   : MPARAM;
      mp2   : MPARAM) : MRESULT [LONG, LOADDS];
   BEGIN
      WITH Settings[comport - COM_OFF] DO
         CASE msg OF
            WM_INITDLG:
               WinSendDlgItemMsg (hwnd, databits, BM_SETCHECK, 1, 0);
               WinSetFocus (HWND_DESKTOP, WinWindowFromID (hwnd, databits));
               RETURN 1;
         |  WM_CONTROL:
               databits := LOWORD (mp1);
               RETURN 0;
         |  WM_COMMAND:
               WinDismissDlg (hwnd, 1);
               RETURN 0;
         ELSE
            RETURN WinDefDlgProc (hwnd, msg, mp1, mp2);
         END;
      END;
   END DataDlgProc;


PROCEDURE StopDlgProc ['StopDlgProc'] (
(* Gère une boîte de dialogue pour choisir 1 ou 2 bits de stop *)
      hwnd  : HWND;
      msg   : USHORT;
      mp1   : MPARAM;
      mp2   : MPARAM) : MRESULT [LONG, LOADDS];
   BEGIN
      WITH Settings[comport - COM_OFF] DO
         CASE msg OF
            WM_INITDLG:
               WinSendDlgItemMsg (hwnd, stopbits, BM_SETCHECK, 1, 0);
               WinSetFocus (HWND_DESKTOP, WinWindowFromID (hwnd, stopbits));
               RETURN 1;
         |  WM_CONTROL:
               stopbits := LOWORD (mp1);
               RETURN 0;
         |  WM_COMMAND:
               WinDismissDlg (hwnd, 1);
               RETURN 0;
         ELSE
            RETURN WinDefDlgProc (hwnd, msg, mp1, mp2);
         END;
      END;
   END StopDlgProc;


PROCEDURE ParityDlgProc ['ParityDlgProc'] (
(* Gère une boîte de dialogue pour choisir la parité *)
      hwnd  : HWND;
      msg   : USHORT;
      mp1   : MPARAM;
      mp2   : MPARAM) : MRESULT [LONG, LOADDS];
   BEGIN
      WITH Settings[comport - COM_OFF] DO
         CASE msg OF
            WM_INITDLG:
               WinSendDlgItemMsg (hwnd, parity, BM_SETCHECK, 1, 0);
               WinSetFocus (HWND_DESKTOP, WinWindowFromID (hwnd, parity));
               RETURN 1;
         |  WM_CONTROL:
               parity := LOWORD (mp1);
               RETURN 0;
         |  WM_COMMAND:
               WinDismissDlg (hwnd, 1);
               RETURN 0;
         ELSE
            RETURN WinDefDlgProc (hwnd, msg, mp1, mp2);
         END;
      END;
   END ParityDlgProc;


PROCEDURE AboutDlgProc ['AboutDlgProc'] (
(* Gère une boîte de dialogue d'info *)
      hwnd  : HWND;
      msg   : USHORT;
      mp1   : MPARAM;
      mp2   : MPARAM) : MRESULT [LONG, LOADDS];
   BEGIN
      IF msg = WM_COMMAND THEN
         WinDismissDlg (hwnd, 1);
         RETURN 0;
      ELSE
         RETURN WinDefDlgProc (hwnd, msg, mp1, mp2);
      END;
   END AboutDlgProc;


PROCEDURE SendFNDlgProc ['SendFNDlgProc'] (
(* Gère une boîte de dialogue pour obtenir le nom du fichier à envoyer *)
      hwnd  : HWND;
      msg   : USHORT;
      mp1   : MPARAM;
      mp2   : MPARAM) : MRESULT [LONG, LOADDS];
   BEGIN
      CASE msg OF
         WM_INITDLG:
            WinSetFocus (HWND_DESKTOP, WinWindowFromID (hwnd, ID_SENDFN));
            RETURN 1;
      |  WM_COMMAND:
            WinQueryDlgItemText (hwnd, ID_SENDFN, 20, ADR (sFname));
            WinDismissDlg (hwnd, 1);
            RETURN 0;
      ELSE
         RETURN WinDefDlgProc (hwnd, msg, mp1, mp2);
      END;
   END SendFNDlgProc;
```

```
PROCEDURE PathDlgProc ['PathDlgProc'] (
(* Gère une boîte de dialogue pour obtenir répertoire *)
      hwnd  : HWND;
      msg   : USHORT;
      mp1   : MPARAM;
      mp2   : MPARAM) : MRESULT [LONG, LOADDS];
   BEGIN
      CASE msg OF
         WM_INITDLG:
          WinSetFocus (HWND_DESKTOP, WinWindowFromID (hwnd, ID_DIRPATH));
          RETURN 1;
      |  WM_COMMAND:
            WinQueryDlgItemText (hwnd, ID_DIRPATH, 60, ADR (Path));
            WinDismissDlg (hwnd, 1);
            RETURN 0;
      ELSE
         RETURN WinDefDlgProc (hwnd, msg, mp1, mp2);
      END;
   END PathDlgProc;


PROCEDURE DirEndDlgProc ['DirEndDlgProc'] (
(* Gère une boîte de dialogue pour annuler un répertoire *)
      hwnd  : HWND;
      msg   : USHORT;
      mp1   : MPARAM;
      mp2   : MPARAM) : MRESULT [LONG, LOADDS];
   BEGIN
      IF msg = WM_COMMAND THEN
         WinDismissDlg (hwnd, 1);
         RETURN 0;
      ELSE
         RETURN WinDefDlgProc (hwnd, msg, mp1, mp2);
      END;
   END DirEndDlgProc;


PROCEDURE HelpDlgProc ['HelpDlgProc'] (
(* Gère une boîte de dialogue d'aide *)
      hwnd  : HWND;
      msg   : USHORT;
      mp1   : MPARAM;
      mp2   : MPARAM) : MRESULT [LONG, LOADDS];
   BEGIN
      IF msg = WM_COMMAND THEN
         WinDismissDlg (hwnd, 1);
         RETURN 0;
      ELSE
         RETURN WinDefDlgProc (hwnd, msg, mp1, mp2);
      END;
   END HelpDlgProc;


PROCEDURE KeyTranslate (mp1, mp2 : MPARAM; VAR c1, c2 : CHAR) : BOOLEAN;
(* Traduit un message WM_CHAR en code ascii *)

   VAR
                           code : CARDINAL;
                           fs : BITSET;
                           VK, KU, CH, CT : BOOLEAN;

   BEGIN
      fs := BITSET (LOWORD (mp1));      (* flags *)
      VK := (fs * BITSET (KC_VIRTUALKEY)) # {};
      KU := (fs * BITSET (KC_KEYUP)) # {};
      CH := (fs * BITSET (KC_CHAR)) # {};
      CT := (fs * BITSET (KC_CTRL)) # {};
      IF (NOT KU) THEN
         code := LOWORD (mp2);         (* code caractère *)
         c1 := CHR (code);
         c2 := CHR (code DIV 256);
         IF ORD (c1) = 0E0H THEN         (* fonction *)
            c1 := 0C;
         END;
         IF CT AND (NOT CH) AND (NOT VK) AND (code # 0) THEN
            c1 := CHR (CARDINAL ((BITSET (ORD (c1)) * BITSET (1FH))));
         END;
         RETURN TRUE;
      ELSE
         RETURN FALSE;
      END;
   END KeyTranslate;


PROCEDURE WindowProc ['WindowProc'] (
(* Procédure de la fenêtre principale *)
(* Traite les messages en provenace de PM et du programme *)
      hwnd  : HWND;
      msg   : USHORT;
      mp1   : MPARAM;
      mp2   : MPARAM) : MRESULT [LONG, LOADDS];

   VAR
      ch : CHAR;
      hps       : HPS;
      pswp      : PSWP;
      c1, c2    : CHAR;

   BEGIN
      CASE msg OF
         WM_HELP:
            IF TermMode THEN
               WinDlgBox (HWND_DESKTOP, hwnd, HelpDlgProc,
                          0, IDM_TERMHELP, 0);
            ELSE
               WinDlgBox (HWND_DESKTOP, hwnd, HelpDlgProc,
                          0, IDM_HELPMENU, 0);
            END;
            RETURN 0;
      |  WM_SETFULL:
            SetFull;
            RETURN 0;
      |  WM_SETRESTORE:
            SetRestore;
            RETURN 0;
      |  WM_SETMAX:
            SetMax;
            RETURN 0;
      |  WM_MINMAXFRAME:
            pswp := PSWP (mp1);
            IF BITSET (pswp^.fs) * BITSET (SWP_MINIMIZE) # {} THEN
               (* n'affiche pas les ports quand fenêtre minimisée *)
               WinSetWindowText (FrameWindow, ADR (Title));
            ELSE
               WinSetWindowText (FrameWindow, ADR (Banner));
               IF TermMode AND
                (BITSET (pswp^.fs) * BITSET (SWP_RESTORE) # {}) THEN
                  (* Force fenêtre maximisée en mode terminal *)
                  WinPostMsg (FrameWindow, WM_SETMAX, 0, 0);
               ELSIF (NOT TermMode) AND
                (BITSET (pswp^.fs) * BITSET (SWP_MAXIMIZE) # {}) THEN
                  (* Fenêtre non maximisée si pas mode terminal *)
                  WinPostMsg (FrameWindow, WM_SETRESTORE, 0, 0);
               ELSE
                  (* ne fait rien *)
               END;
            END;
            RETURN WinDefWindowProc (hwnd, msg, mp1, mp2);
      |  WM_CREATE:
          hdc := WinOpenWindowDC (hwnd);
          VioCreatePS (frame_hvps, 25, 80, 0, FORMAT_CGA, 0);
          VioCreatePS (child_hvps, 16, 40, 0, FORMAT_CGA, 0);
          DosCreateThread (TermThrProc, TermThr, ADR (TermStack[STKSIZE]));
          DosSuspendThread (TermThr);
          RETURN 0;
      |  WM_INITMENU:
            Check (ColorSet);
            RETURN 0;
      |  WM_COMMAND:
            DoMenu (hwnd, mp1);
            RETURN 0;
      |  WM_TERMQUIT:
            TermMode := FALSE;
            DosSuspendThread (TermThr);
            VioAssociate (0, hvps);
            (* Restore la taille de la fenêtre *)
            SetRestore;
            Enable (IDM_CONNECT);
            Enable (IDM_SEND);
            Enable (IDM_REC);
            Enable (IDM_DIR);
            Enable (IDM_OPTIONS);
            Enable (IDM_COLORS);
            RETURN 0;
      |  WM_TERM:
            PutPortChar (CHR (LOWORD (mp1)));    (* à l'écran *)
            RETURN 0;
      |  WM_CHAR:
            IF TermMode THEN
               IF KeyTranslate (mp1, mp2, c1, c2) THEN
                  PutKbdChar (c1, c2);    (* To Port *)
                  RETURN 0;
               ELSE
                  RETURN WinDefAVioWindowProc (hwnd, msg, mp1, mp2);
               END;
            ELSE
               RETURN WinDefWindowProc (hwnd, msg, mp1, mp2);
            END;
      |  WM_PAINT:
            hps := WinBeginPaint (hwnd, NULL, ADDRESS (NULL));
            GpiErase (hps);
            VioShowPS (25, 80, 0, hvps);
            WinEndPaint (hps);
            RETURN 0;
      |  WM_SIZE:
            IF TermMode THEN
               RETURN WinDefAVioWindowProc (hwnd, msg, mp1, mp2);
            ELSE
               RETURN WinDefWindowProc (hwnd, msg, mp1, mp2);
            END;
      |  WM_DESTROY:
            VioDestroyPS (frame_hvps);
            VioDestroyPS (child_hvps);
            RETURN 0;
      ELSE
         RETURN WinDefWindowProc (hwnd, msg, mp1, mp2);
      END;
   END WindowProc;


PROCEDURE ChildWindowProc ['ChildWindowProc'] (
   (* Procédure de fenêtre pour les fenêtres enfant
      d'émission et de réception *)
   hwnd : HWND;
   msg  : USHORT;
   mp1  : MPARAM;
   mp2  : MPARAM) : MRESULT [LONG, LOADDS];

   VAR
```

```
      mp : USHORT;
      hps : HPS;
      c1, c2 : CHAR;

   BEGIN
      CASE msg OF
         WM_PAINT:
            hps := WinBeginPaint (hwnd, NULL, ADDRESS (NULL));
            GpiErase (hps);
            VioShowPS (16, 40, 0, hvps);
            WinEndPaint (hps);
            RETURN 0;
      | WM_CHAR:
            IF KeyTranslate (mp1, mp2, c1, c2) AND (c1 = ESC) THEN
               Aborted := TRUE;
               RETURN 0;
            ELSE
               RETURN WinDefWindowProc (hwnd, msg, mp1, mp2);
            END;
      | WM_PAD:
            mp := LOWORD (mp1);
            IF (mp = PAD_Error) OR (mp = PAD_Quit) THEN
               WriteLn;
               IF mp = PAD_Error THEN
                  WinMessageBox (HWND_DESKTOP, hwnd,
                          ADR ("Transfert interrompu"),
                          ADR (Class), 0, MB_OK + MB_ICONEXCLAMATION);
               ELSE
                  WinMessageBox (HWND_DESKTOP, hwnd,
                              ADR ("Transfert terminé"),
                              ADR (Class), 0, MB_OK + MB_ICONASTERISK);
               END;
               DosSleep (2000);
               VioAssociate (0, hvps);
               WinDestroyWindow(ChildFrameWindow);
               Enable (IDM_CONNECT);
               Enable (IDM_SEND);
               Enable (IDM_REC);
               Enable (IDM_DIR);
               Enable (IDM_OPTIONS);
               Enable (IDM_COLORS);
            ELSE
               DoPADMsg (mp1, mp2);
            END;
            RETURN 0;
      | WM_DL:
            DoDLMsg (mp1, mp2);
            RETURN 0;
      | WM_SIZE:
            WinSetWindowPos (ChildFrameWindow, 0,
               Pos.cx DIV 4, Pos.cy DIV 4,
               Pos.cx DIV 2, Pos.cy DIV 2 - 3,
               SWP_MOVE + SWP_SIZE);
            RETURN WinDefAVioWindowProc (hwnd, msg, mp1, mp2);
      ELSE
         RETURN WinDefWindowProc (hwnd, msg, mp1, mp2);
      END;
   END ChildWindowProc;


BEGIN   (* Module d'initialisation *)
   WITH Settings[ID_COM1 - COM_OFF] DO
      baudrate := ID_B1200;
      parity := ID_EVEN;
      databits := ID_DATA7;
      stopbits := ID_STOP1;
   END;

   WITH Settings[ID_COM2 - COM_OFF] DO
      baudrate := ID_B19K2;
      parity := ID_EVEN;
      databits := ID_DATA7;
      stopbits := ID_STOP1;
   END;
   PrevComPort := NONE;
   comport := ID_COM1;
   TermMode := FALSE;   (* Pas en mode terminal par défaut *)
END Shell.
```

[LISTING DIX]

```
IMPLEMENTATION MODULE Term;   (* Emulation de terminal TVI 950 *)

   FROM Drives IMPORT
      SetDrive;

   FROM Directories IMPORT
      FileAttributes, AttributeSet, DirectoryEntry, FindFirst, FindNext;

   FROM SYSTEM IMPORT
      ADR;

   FROM OS2DEF IMPORT
      ULONG;

   FROM DosCalls IMPORT
      DosChDir, DosSleep;

   FROM Screen IMPORT
      ClrScr, ClrEol, GotoXY, GetXY,
      Right, Left, Up, Down, WriteAtt, WriteString, WriteLn, Write,
      attribute, NORMAL, HIGHLIGHT, REVERSE;

   FROM PMWIN IMPORT
      WinPostMsg, MPFROM2SHORT;

   FROM Shell IMPORT
      comport, FrameWindow;

   FROM KH IMPORT
      COM_OFF;

   FROM CommPort IMPORT
      CommStatus, GetChar, SendChar;

   FROM Strings IMPORT
      Length, Concat;

   IMPORT ASCII;


   CONST
      (* Codes touches :  Note: F1 — F12 sont
         en fait Shift-F1 — Shift-F12 *)
      F1 = 124C;
      F2 = 125C;
      F3 = 126C;
      F4 = 127C;
      F5 = 130C;
      F6 = 131C;
      F7 = 132C;
      F8 = 133C;
      F9 = 134C;
      F10 = 135C;
      F11 = 207C;
      F12 = 210C;
      AF1 = 213C;    (* Alt-F1 *)
      AF2 = 214C;    (* Alt-F2 *)
      INS = 122C;
      DEL = 123C;
      HOME = 107C;
      PGDN = 121C;    (* synonyme de PF10 *)
      PGUP = 111C;    (* synonyme de PF11 *)
      ENDD = 117C;    (* synonyme de PF12 *)
      UPARROW = 110C;
      DOWNARROW = 120C;
      LEFTARROW = 113C;
      RIGHTARROW = 115C;
      CtrlX = 30C;
      CtrlCaret = 36C;
      CtrlZ = 32C;
      CtrlL = 14C;
      CtrlH = 10C;
      CtrlK = 13C;
      CtrlJ = 12C;
      CtrlV = 26C;
      ESC = 33C;
      BUFSIZE = 4096;    (* buffer caractères utilisé par la tâche term *)


   VAR
      commStat : CommStatus;
      echo : (Off, Local, On);
      newline: BOOLEAN;    (* traduit <cr> en <cr><lf> *)
      Insert : BOOLEAN;


   PROCEDURE Dir (path : ARRAY OF CHAR);
   (* Change drive et/ou directory; affiche un directory *)

      VAR
         gotFN : BOOLEAN;
         filename : ARRAY [0..20] OF CHAR;
         attr : AttributeSet;
         ent : DirectoryEntry;
         i, j, k : INTEGER;

      BEGIN
         filename := "";    (* au cas où on ne change pas de directory *)
         i := Length (path);
         IF (i > 2) AND (path[1] = ':') THEN    (* drive *)
            DEC (i, 2);
            SetDrive (ORD (CAP (path[0])) - ORD ('A'));
            FOR j := 0 TO i DO    (* supprime le drive *)
               path[j] := path[j + 2];
            END;
         END;
         IF i # 0 THEN
            gotFN := FALSE;
            WHILE (i >= 0) AND (path[i] # '\') DO
               IF path[i] = '.' THEN
                  gotFN := TRUE;
               END;
               DEC (i);
            END;
            IF gotFN THEN
               j := i + 1;
               k := 0;
               WHILE path[j] # 0C DO
                  filename[k] := path[j];
                  INC (k);       INC (j);
               END;
               filename[k] := 0C;
               IF (i = -1) OR ((i = 0) AND (path[0] = '\')) THEN
                  INC (i);
               END;
               path[i] := 0C;
            END;
```

```
      END;
      IF Length (path) # 0 THEN
         DosChDir (ADR (path), 0);
      END;
      IF Length (filename) = 0 THEN
         filename := "*.*";
      END;
      attr := AttributeSet {ReadOnly, Directory, Archive};
      i := 1;   (* position dans la line *)

      ClrScr;
      gotFN := FindFirst (filename, attr, ent);
      WHILE gotFN DO
         WriteString (ent.name);
         j := Length (ent.name);
         WHILE j < 12 DO   (* 12 = longueur max d'un nom de fichier *)
            Write (' ');
            INC (j);
         END;
         INC (i);   (* position suivante dans la ligne *)
         IF i > 5 THEN
            i := 1;   (* recommence sur une nouvelle ligne *)
            WriteLn;
         ELSE
            WriteString (" | ");
         END;
         gotFN := FindNext (ent);
      END;
      WriteLn;
   END Dir;


PROCEDURE InitTerm;
(* Efface l'écran, prépare l'émulation terminal *)
   BEGIN
      ClrScr;
      Insert := FALSE;
      attribute := NORMAL;
   END InitTerm;


PROCEDURE PutKbdChar (ch1, ch2 : CHAR);
(* Traite un caractère lu au clavier *)
   BEGIN
      IF ch1 = ASCII.enq THEN   (* Control-E *)
         echo := On;
      ELSIF ch1 = ASCII.ff THEN   (* Control-L *)
         echo := Local;
      ELSIF ch1 = ASCII.dc4 THEN   (* Control-T *)
         echo := Off;
      ELSIF ch1 = ASCII.so THEN   (* Control-N *)
         newline := TRUE;
      ELSIF ch1 = ASCII.si THEN   (* Control-O *)
         newline := FALSE;
      ELSIF (ch1 = ASCII.can) OR (ch1 = ESC) THEN
         attribute := NORMAL;
         WinPostMsg (FrameWindow, WM_TERMQUIT, 0, 0);
      ELSIF ch1 = 0C THEN
         Function (ch2);
      ELSE
         commStat := SendChar (comport - COM_OFF, ch1, FALSE);
         IF (echo = On) OR (echo = Local) THEN
            WriteAtt (ch1);
         END;
      END;
   END PutKbdChar;


PROCEDURE Function (ch : CHAR);
(* Traite les touches de fonction — y compris PF1 - PF12, etc. *)
   BEGIN
      CASE ch OF
         F1 :  commStat := SendChar (comport - COM_OFF, ASCII.soh, FALSE);
               commStat := SendChar (comport - COM_OFF, '@', FALSE);
               commStat := SendChar (comport - COM_OFF, ASCII.cr, FALSE);
       | F2 :  commStat := SendChar (comport - COM_OFF, ASCII.soh, FALSE);
               commStat := SendChar (comport - COM_OFF, 'A', FALSE);
               commStat := SendChar (comport - COM_OFF, ASCII.cr, FALSE);
       | F3 :  commStat := SendChar (comport - COM_OFF, ASCII.soh, FALSE);
               commStat := SendChar (comport - COM_OFF, 'B', FALSE);
               commStat := SendChar (comport - COM_OFF, ASCII.cr, FALSE);
       | F4 :  commStat := SendChar (comport - COM_OFF, ASCII.soh, FALSE);
               commStat := SendChar (comport - COM_OFF, 'C', FALSE);
               commStat := SendChar (comport - COM_OFF, ASCII.cr, FALSE);
       | F5 :  commStat := SendChar (comport - COM_OFF, ASCII.soh, FALSE);
               commStat := SendChar (comport - COM_OFF, 'D', FALSE);
               commStat := SendChar (comport - COM_OFF, ASCII.cr, FALSE);
       | F6 :  commStat := SendChar (comport - COM_OFF, ASCII.soh, FALSE);
               commStat := SendChar (comport - COM_OFF, 'E', FALSE);
               commStat := SendChar (comport - COM_OFF, ASCII.cr, FALSE);
       | F7 :  commStat := SendChar (comport - COM_OFF, ASCII.soh, FALSE);
               commStat := SendChar (comport - COM_OFF, 'F', FALSE);
               commStat := SendChar (comport - COM_OFF, ASCII.cr, FALSE);
       | F8 :  commStat := SendChar (comport - COM_OFF, ASCII.soh, FALSE);
               commStat := SendChar (comport - COM_OFF, 'G', FALSE);
               commStat := SendChar (comport - COM_OFF, ASCII.cr, FALSE);
       | F9 :  commStat := SendChar (comport - COM_OFF, ASCII.soh, FALSE);
               commStat := SendChar (comport - COM_OFF, 'H', FALSE);
               commStat := SendChar (comport - COM_OFF, ASCII.cr, FALSE);
       | F10,
         PGDN: commStat := SendChar (comport - COM_OFF, ASCII.soh, FALSE);
               commStat := SendChar (comport - COM_OFF, 'I', FALSE);
               commStat := SendChar (comport - COM_OFF, ASCII.cr, FALSE);
       | F11,
         AF1,
         PGUP: commStat := SendChar (comport - COM_OFF, ASCII.soh, FALSE);
               commStat := SendChar (comport - COM_OFF, 'J', FALSE);
               commStat := SendChar (comport - COM_OFF, ASCII.cr, FALSE);
       | F12,
         AF2,
         ENDD: commStat := SendChar (comport - COM_OFF, ESC, FALSE);
               commStat := SendChar (comport - COM_OFF, 'Q', FALSE);
       | INS : IF NOT Insert THEN
                  commStat := SendChar (comport - COM_OFF, ESC, FALSE);
                  commStat := SendChar (comport - COM_OFF, 'E', FALSE);
               END;
       | DEL : commStat := SendChar (comport - COM_OFF, ESC, FALSE);
               commStat := SendChar (comport - COM_OFF, 'R', FALSE);
       | HOME      : commStat := SendChar (comport - COM_OFF, CtrlZ, FALSE);
       | UPARROW   : commStat := SendChar (comport - COM_OFF, CtrlK, FALSE);
       | DOWNARROW : commStat := SendChar (comport - COM_OFF, CtrlV, FALSE);
       | LEFTARROW : commStat := SendChar (comport - COM_OFF, CtrlH, FALSE);
       | RIGHTARROW: commStat := SendChar (comport - COM_OFF, CtrlL, FALSE);
       ELSE
          (* ne fait rien *)
       END;
   END Function;


PROCEDURE TermThrProc;
(* Tâche qui lit les caractères dans le port série *)
(* et les écrit dans un buffer *)

   VAR
      ch : CHAR;

   BEGIN
      LOOP
         IF GetChar (comport - COM_OFF, ch) = Success THEN
          WinPostMsg (FrameWindow, WM_TERM, MPFROM2SHORT (ORD (ch), 0), 0);
         ELSE
            DosSleep (0);
         END
      END;
   END TermThrProc;


VAR
   EscState, CurState1, CurState2 : BOOLEAN;
   CurChar1 : CHAR;

PROCEDURE PutPortChar (ch : CHAR);
(* Traite un caractère lu dans un port *)
   BEGIN
      IF EscState THEN
         EscState := FALSE;
         IF ch = '=' THEN
            CurState1 := TRUE;
         ELSE
            Escape (ch);
         END;
      ELSIF CurState1 THEN
         CurState1 := FALSE;
         CurChar1 := ch;
         CurState2 := TRUE;
      ELSIF CurState2 THEN
         CurState2 := FALSE;
         Cursor (ch);
      ELSE
         CASE ch OF
            CtrlCaret, CtrlZ : ClrScr;
          | CtrlL : Right;
          | CtrlH : Left;
          | CtrlK : Up;
          | CtrlJ : Down;
          | ESC   : EscState := TRUE;
         ELSE
            WriteAtt (ch);
            IF newline AND (ch = ASCII.cr) THEN
               WriteLn;
            END;
         END;
      END;
      IF echo = On THEN
         commStat := SendChar (comport - COM_OFF, ch, FALSE);
      END;
   END PutPortChar;


PROCEDURE Escape (ch : CHAR);
(* traite les séquences d'échappement *)
   BEGIN
      CASE ch OF
         '*' : ClrScr;
       | 'T', 'R' : ClrEol;
       | ')' : attribute := NORMAL;
       | '(' : attribute := HIGHLIGHT;
       | 'f' : InsertMsg;
       | 'g' : InsertOn;
      ELSE
         (* ignore *)
      END;
   END Escape;


PROCEDURE Cursor (ch : CHAR);
(* Gère le positionnement du curseur *)

   VAR
      x, y : CARDINAL;
```

```
    BEGIN
       y := ORD (CurChar1) - 20H;
       x := ORD (ch) - 20H;
       GotoXY (x, y);   (* ajuste pour que HOME soit égal à (1, 1) *)
    END Cursor;


  VAR
     cx, cy : CARDINAL;

  PROCEDURE InsertMsg;
  (* mode insertion — affiche un message en bas de l'écran *)
     BEGIN
        IF NOT Insert THEN
           GetXY (cx, cy);    (* sauve position courante *)
           GotoXY (1, 24);
           ClrEol;
           attribute := REVERSE;
        ELSE    (* quitte mode insertion *)
           GetXY (cx, cy);
           GotoXY (1, 24);
           ClrEol;
           GotoXY (cx, cy);
           Insert := FALSE;
        END;
     END InsertMsg;


  PROCEDURE InsertOn;
  (* entre en mode insertion — après que le message ait été écrit *)
     BEGIN
        attribute := NORMAL;
        GotoXY (cx, cy);
        Insert := TRUE;
     END InsertOn;


BEGIN   (* initialisation du module *)
   echo := Off;
   newline := FALSE;
   Insert := FALSE;
   EscState := FALSE;
   CurState1 := FALSE;
   CurState2 := FALSE;
END Term.


[LISTING ONZE]

IMPLEMENTATION MODULE Screen;
(* module de gestion d'écran de bas niveau (via AVIO) *)

   IMPORT ASCII;

   FROM SYSTEM IMPORT
      ADR;

   FROM Strings IMPORT
      Length;

   FROM Conversions IMPORT
      IntToString;

   FROM KH IMPORT
      IDM_GREEN;

   FROM Vio IMPORT
      VioSetCurPos, VioGetCurPos, VioScrollUp,
      VioWrtNCell, VioWrtTTY, VioCell;


   CONST
      GREY = 07H;
      WHITE = 0FH;
      REV_GY = 70H;
      GREEN = 02H;
      LITE_GRN = 0AH;
      REV_GRN = 20H;
      AMBER = 06H;
      LITE_AMB = 0EH;
      REV_AMB = 60H;
      RED = 0CH;
      CY_BK = 0B0H;
      CY_BL = 0B9H;
      REV_RD = 0CFH;
      REV_BL = 9FH;
      MAGENTA = 05H;


   VAR
      (* Dans le module Definition
      NORMAL : CARDINAL;
      HIGHLIGHT : CARDINAL;
      REVERSE : CARDINAL;
                attribute : CARDINAL;
      hvps : HVPS;
      *)
            x, y : CARDINAL;
            bCell : VioCell;


   PROCEDURE White;
   (* Initialise couleurs : Monochrome blanc *)
      BEGIN
         NORMAL := GREY;
         HIGHLIGHT := WHITE;
         REVERSE := REV_GY;
         attribute := NORMAL;
      END White;


   PROCEDURE Green;
   (* Initialise couleurs : Monochrome vert *)
      BEGIN
         NORMAL := GREEN;
         HIGHLIGHT := LITE_GRN;
         REVERSE := REV_GRN;
         attribute := NORMAL;
      END Green;


   PROCEDURE Amber;
   (* Initialise couleurs : Monochrome ambre *)
      BEGIN
         NORMAL := AMBER;
         HIGHLIGHT := LITE_AMB;
         REVERSE := REV_AMB;
         attribute := NORMAL;
      END Amber;


   PROCEDURE Color1;
   (* Initialise couleurs : bleu, rouge, vert *)
      BEGIN
         NORMAL := GREEN;
         HIGHLIGHT := RED;
         REVERSE := REV_BL;
         attribute := NORMAL;
      END Color1;


   PROCEDURE Color2;
   (* Initialise couleurs : fond cyan; noir, bleu, blanc_sur-rouge *)
      BEGIN
         NORMAL := CY_BK;
         HIGHLIGHT := CY_BL;
         REVERSE := REV_RD;
         attribute := NORMAL;
      END Color2;


   PROCEDURE HexToString (num : INTEGER;
                          size : CARDINAL;
                          VAR buf : ARRAY OF CHAR;
                          VAR I : CARDINAL;
                          VAR Done : BOOLEAN);
   (* Procedure locale de conversion entier -> nombre hexa *)

      CONST
         ZERO = 30H;    (* ASCII code *)
         A = 41H;

      VAR
         i : CARDINAL;
         h : CARDINAL;
         t : ARRAY [0..10] OF CHAR;

      BEGIN
         i := 0;
         REPEAT
            h := num MOD 16;
            IF h <= 9 THEN
               t[i] := CHR (h + ZERO);
            ELSE
               t[i] := CHR (h - 10 + A);
            END;
            INC (i);
            num := num DIV 16;
         UNTIL num = 0;

         IF (size > HIGH (buf)) OR (i > HIGH (buf)) THEN
            Done := FALSE;
            RETURN;
         ELSE
            Done := TRUE;
         END;

         WHILE size > i DO
            buf[I] := '0';    (* remplis avec des zeros *)
            DEC (size);
            INC (I);
         END;

         WHILE i > 0 DO
            DEC (i);
            buf[I] := t[i];
            INC (I);
         END;

         buf[I] := 0C;
      END HexToString;


   PROCEDURE ClrScr;
   (* Efface l'écran *)
      BEGIN
         bCell.ch := ' ';        (* espace = écran blanc *)
         bCell.attr := CHR (NORMAL);      (* Attribut vidéo normale *)
         VioScrollUp (0, 0, 24, 79, 25, bCell, hvps);
         GotoXY (0, 0);
```

```
END ClrScr;


PROCEDURE ClrEol;
(* Efface lafin de la ligne courante *)
   BEGIN
      GetXY (x, y);      (* position du curseur courante *)
      bCell.ch := ' ';     (* espacé = blanc *)
      bCell.attr := CHR (NORMAL);    (* Attribut vidéo normale *)
      VioScrollUp (y, x, y, 79, 1, bCell, hvps);
   END ClrEol;


PROCEDURE Right;
(* déplace curseur vers la droite *)
   BEGIN
      GetXY (x, y);
      INC (x);
      GotoXY (x, y);
   END Right;


PROCEDURE Left;
(* déplace curseur vers la gauche *)
   BEGIN
      GetXY (x, y);
      DEC (x);
      GotoXY (x, y);
   END Left;


PROCEDURE Up;
(* déplace curseur vers le haut *)
   BEGIN
      GetXY (x, y);
      DEC (y);
      GotoXY (x, y);
   END Up;


PROCEDURE Down;
(* déplace curseur vers le bas *)
   BEGIN
      GetXY (x, y);
      INC (y);
      GotoXY (x, y);
   END Down;


PROCEDURE GotoXY (col, row : CARDINAL);
(* positionne le curseur *)
   BEGIN
      IF (col <= 79) AND (row <= 24) THEN
         VioSetCurPos (row, col, hvps);
      END;
   END GotoXY;


PROCEDURE GetXY (VAR col, row : CARDINAL);
(* retourne la position du curseur *)
   BEGIN
      VioGetCurPos (row, col, hvps);
   END GetXY;


PROCEDURE Write (c : CHAR);
(* Ecrit un caractère *)
   BEGIN
      WriteAtt (c);
   END Write;


PROCEDURE WriteString (str : ARRAY OF CHAR);
(* Ecrit une chaîne *)

   VAR
      i : CARDINAL;
      c : CHAR;

   BEGIN
      i := 0;
      c := str[i];
      WHILE c # 0C DO
         Write (c);
         INC (i);
         c := str[i];
      END;
   END WriteString;


PROCEDURE WriteInt (n : INTEGER; s : CARDINAL);
(* Ecrit un nombre entier *)

   VAR
      i : CARDINAL;
      b : BOOLEAN;
      str : ARRAY [0..6] OF CHAR;

   BEGIN
      i := 0;
      IntToString (n, s, str, i, b);
      WriteString (str);
   END WriteInt;


PROCEDURE WriteHex (n, s : CARDINAL);
(* Ecrit un nombre hexa *)

   VAR
      i : CARDINAL;
      b : BOOLEAN;
      str : ARRAY [0..6] OF CHAR;

   BEGIN
      i := 0;
      HexToString (n, s, str, i, b);
      WriteString (str);
   END WriteHex;


PROCEDURE WriteLn;
(* Ecrit <cr> <lf> *)
   BEGIN
      Write (ASCII.cr);   Write (ASCII.lf);
   END WriteLn;


PROCEDURE WriteAtt (c : CHAR);
(* Ecrit un caractère et ses attributs *)

   VAR
      s : ARRAY [0..1] OF CHAR;

   BEGIN
      GetXY (x, y);
      IF (c = ASCII.ht) THEN
         bCell.ch := ' ';
         bCell.attr := CHR (attribute);
         REPEAT
            VioWrtNCell (bCell, 1, y, x, hvps);
            Right;
         UNTIL (x MOD 8) = 0;
      ELSIF (c = ASCII.cr) OR (c = ASCII.lf)
       OR (c = ASCII.bel) OR (c = ASCII.bs) THEN
         s[0] := c;    s[1] := 0C;
         VioWrtTTY (ADR (s), 1, hvps);
         IF c = ASCII.lf THEN
            ClrEol;
         END;
      ELSE
         bCell.ch := c;
         bCell.attr := CHR (attribute);
         VioWrtNCell (bCell, 1, y, x, hvps);
         Right;
      END;
   END WriteAtt;

BEGIN     (* initialisation du module *)
   ColorSet := IDM_GREEN;
   NORMAL := GREEN;
   HIGHLIGHT := LITE_GRN;
   REVERSE := REV_GRN;
   attribute := NORMAL;
END Screen.
```

[LISTING DOUZE]

```
(*************************************************************************)
(*                                                                       *)
(*                    Copyright (c) 1988, 1989                           *)
(*                      Stony Brook Software                             *)
(*                              et                                       *)
(*                       Copyright (c) 1990                              *)
(*                       Brian R. Anderson                               *)
(*                      Tous droits réservés                             *)
(*                                                                       *)
(*************************************************************************)

IMPLEMENTATION MODULE CommPort [7];

   FROM SYSTEM IMPORT
      ADR, BYTE, WORD, ADDRESS;

   FROM Storage IMPORT
      ALLOCATE, DEALLOCATE;

   FROM DosCalls IMPORT
      DosOpen, AttributeSet, DosDevIOCtl, DosClose, DosRead, DosWrite;


   TYPE
      CP = POINTER TO CHAR;

   VAR
      pn : CARDINAL;
      Handle : ARRAY [0..3] OF CARDINAL;
      BufIn : ARRAY [0..3] OF CP;
      BufOut : ARRAY [0..3] OF CP;
      BufStart : ARRAY [0..3] OF CP;
      BufLimit : ARRAY [0..3] OF CP;
      BufSize : ARRAY [0..3] OF CARDINAL;
      Temp : ARRAY [1..1024] OF CHAR;    (* taille de la queue OS/2 série *)


   PROCEDURE CheckPort (portnum : CARDINAL) : BOOLEAN;
   (* Cherche un numéro de port valide, et l'ouvre si ce n'est déjà fait *)

      CONST
```

```
      PortName : ARRAY [0..3] OF ARRAY [0..4] OF CHAR =
        [['COM1', 0C], ['COM2', 0C], ['COM3', 0C], ['COM4', 0C]];

   VAR
      Action : CARDINAL;

   BEGIN
      (* Teste no de port *)
      IF portnum > 3 THEN
         RETURN FALSE;
      END;

      (* ouvre le port si ce n'est déjà fait *)
      IF Handle[portnum] = 0 THEN
         IF DosOpen(ADR(PortName[portnum]), Handle[portnum], Action, 0,
          AttributeSet{}, 1, 12H, 0) # 0 THEN
            RETURN FALSE;
         END;
      END;
      RETURN TRUE;
   END CheckPort;



PROCEDURE InitPort (portnum : CARDINAL; speed : BaudRate; data : DataBits;
                    stop : StopBits; check : Parity) : CommStatus;
(* Initialise un port *)

   CONST
      Rate : ARRAY BaudRate OF CARDINAL =
               [110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200];
      TransParity : ARRAY Parity OF BYTE = [2, 1, 0];

   TYPE
      LineChar =  RECORD
                     bDataBits : BYTE;
                     bParity : BYTE;
                     bStopBits : BYTE;
                  END;

   VAR
      LC : LineChar;

   BEGIN
      (* Teste numéro de port *)
      IF NOT CheckPort(portnum) THEN
         RETURN InvalidPort;
      END;

      (* Modifie vitesse en bauds *)
   IF DosDevIOCtl(0, ADR(Rate[speed]), 41H, 1, Handle[portnum]) # 0 THEN
         RETURN InvalidParameter;
   END;

      (* modifie caractéristiques *)
      LC.bDataBits := BYTE(data);
      IF stop = 1 THEN
         DEC (stop);     (* 0x00 = 1 stop bits;     0x02 = 2 stop bits *)
      END;
      LC.bStopBits := BYTE(stop);
      LC.bParity := TransParity[check];

      IF DosDevIOCtl(0, ADR(LC), 42H, 1, Handle[portnum]) # 0 THEN
         RETURN InvalidParameter;
      END;

      RETURN Success;
   END InitPort;


PROCEDURE StartReceiving (portnum, bufsize : CARDINAL) : CommStatus;
(* Commence la réception sur un port *)
   BEGIN
      IF NOT CheckPort(portnum) THEN
         RETURN InvalidPort;
      END;
      IF BufStart[portnum] # NIL THEN
         RETURN AlreadyReceiving;
      END;
      ALLOCATE (BufStart[portnum], bufsize);
      BufIn[portnum] := BufStart[portnum];
      BufOut[portnum] := BufStart[portnum];
      BufLimit[portnum] := BufStart[portnum];
      INC (BufLimit[portnum]:ADDRESS, bufsize - 1);
      BufSize[portnum] := bufsize;
      RETURN Success;
   END StartReceiving;


PROCEDURE StopReceiving (portnum : CARDINAL) : CommStatus;
(* Stoppe la réception sur un port *)
   BEGIN
      IF NOT CheckPort(portnum) THEN
         RETURN InvalidPort;
      END;
      IF BufStart[portnum] # NIL THEN
         DEALLOCATE (BufStart[portnum], BufSize[portnum]);
         BufLimit[portnum] := NIL;
         BufIn[portnum] := NIL;
         BufOut[portnum] := NIL;
         BufSize[portnum] := 0;
      END;
      DosClose(Handle[portnum]);
      Handle[portnum] := 0;
      RETURN Success;
   END StopReceiving;


   PROCEDURE GetChar (portnum : CARDINAL; VAR ch : CHAR) : CommStatus;
   (* Lit un caractère sur un port *)

      VAR
         status : CARDINAL;
         read : CARDINAL;
         que : RECORD
                  ct : CARDINAL;
                  sz : CARDINAL;
               END;
         i : CARDINAL;

      BEGIN
         IF BufStart[portnum] = NIL THEN
            RETURN NotReceiving;
         END;
         IF NOT CheckPort(portnum) THEN
            RETURN InvalidPort;
         END;
         status := DosDevIOCtl (ADR (que), 0, 68H, 1, Handle[portnum]);
         IF (status = 0) AND (que.ct # 0) THEN
            status := DosRead (Handle[portnum], ADR (Temp), que.ct, read);
            IF (status # 0) OR (read = 0) THEN
               RETURN NotReceiving;
            END;
            FOR i := 1 TO read DO
               BufIn[portnum]^ := Temp[i];
               IF BufIn[portnum] = BufLimit[portnum] THEN
                  BufIn[portnum] := BufStart[portnum];
               ELSE
                  INC (BufIn[portnum]:ADDRESS);
               END;
               IF BufIn[portnum] = BufOut[portnum] THEN
                  RETURN BufferOverflow;
               END;
            END;
         END;

         IF BufIn[portnum] = BufOut[portnum] THEN
            RETURN NoCharacter;
         END;
         ch := BufOut[portnum]^;
         IF BufOut[portnum] = BufLimit[portnum] THEN
            BufOut[portnum] := BufStart[portnum];
         ELSE
            INC (BufOut[portnum]:ADDRESS);
         END;
         RETURN Success;
      END GetChar;


   PROCEDURE SendChar (portnum : CARDINAL; ch : CHAR;
                       modem : BOOLEAN) : CommStatus;
   (* Envoie un caractère sur un port *)

      VAR
         wrote : CARDINAL;
         status : CARDINAL;
         commSt : CHAR;

      BEGIN
         IF NOT CheckPort(portnum) THEN
            RETURN InvalidPort;
         END;
         status := DosDevIOCtl (ADR (commSt), 0, 64H, 1, Handle[portnum]);
         IF (status # 0) OR (commSt # 0C) THEN
            RETURN TimeOut;
         ELSE
            status := DosWrite(Handle[portnum], ADR(ch), 1, wrote);
            IF (status # 0) OR (wrote # 1) THEN
               RETURN TimeOut;
            ELSE
               RETURN Success;
            END;
         END;
      END SendChar;


BEGIN   (* initialisation du module *)
   (* encore rien d'ouvert *)
   FOR pn := 0 TO 3 DO
      Handle[pn] := 0;
      BufStart[pn] := NIL;
      BufLimit[pn] := NIL;
      BufIn[pn] := NIL;
      BufOut[pn] := NIL;
      BufSize[pn] := 0;
   END;
END CommPort.


[LISTING TREIZE]

IMPLEMENTATION MODULE Files;    (* E/S fichier pour Kermit *)

   FROM FileSystem IMPORT
      File, Response, Delete, Lookup, Close, ReadNBytes, WriteNBytes;

   FROM Strings IMPORT
      Append;

   FROM Conversions IMPORT
```

```
    CardToString;

FROM SYSTEM IMPORT
   ADR, SIZE;


TYPE
   buffer = ARRAY [1..512] OF CHAR;


VAR
   ext : CARDINAL;
   (* nouvelle extension de fichier pour éviter les conflits de nom *)
   inBuf, outBuf : buffer;
   inP, outP : CARDINAL;    (* pointeurs buffers *)
   read, written : CARDINAL;    (* nombre d'octets lus ou écrits *)
                                (* par ReadNBytes ou WriteNBytes *)


PROCEDURE Open (VAR f : File; name : ARRAY OF CHAR) : Status;
(* ouvre un fichier existant en lecture, retourne status *)
   BEGIN
      Lookup (f, name, FALSE);
      IF f.res = done THEN
         inP := 0;   read := 0;
         RETURN Done;
      ELSE
         RETURN Error;
      END;
   END Open;


PROCEDURE Create (VAR f : File; name : ARRAY OF CHAR) : Status;
(* Crée un nouveau fichier pour écriture, retourne status *)

   VAR
      ch : CHAR;
      str : ARRAY [0..3] OF CHAR;
      i : CARDINAL;
      b : BOOLEAN;

   BEGIN
      LOOP
         Lookup (f, name, FALSE);    (* teste si le fichier existe *)
         IF f.res = done THEN
            Close (f);
            (* le fichier existe : change le nom *)
            IF ext > 99 THEN    (* il n'y a plus de noms... *)
               RETURN Error;
            END;
            i := 0;
            WHILE (name[i] # 0C) AND (name[i] # '.') DO
               INC (i);    (* cherche la fin du fichier *)
            END;
            name[i] := '.';    name[i + 1] := 'K';    name[i + 2] := 0C;
            i := 0;
            CardToString (ext, 1, str, i, b);
            Append (name, str);    (* ajoute l'extension *)
            INC (ext);
         ELSE
            EXIT;
         END;
      END;
      Lookup (f, name, TRUE);
      IF f.res = done THEN
         outP := 0;
         RETURN Done;
      ELSE
         RETURN Error;
      END;
   END Create;


PROCEDURE CloseFile (VAR f : File; Which : FileType) : Status;
(* ferme un fichier après lecture ou écriture *)
   BEGIN
      written := outP;
      IF (Which = Output) AND (outP > 0) THEN
         WriteNBytes (f, ADR (outBuf), outP);
         written := f.count;
      END;
      Close (f);
      IF (written = outP) AND (f.res = done) THEN
         RETURN Done;
      ELSE
         RETURN Error;
      END;
   END CloseFile;


PROCEDURE Get (VAR f : File; VAR ch : CHAR) : Status;
(* Lit un caractère dans un fichier, retourne status *)
   BEGIN
      IF inP = read THEN
         ReadNBytes (f, ADR (inBuf), SIZE (inBuf));
         read := f.count;
         inP := 0;
      END;
      IF read = 0 THEN
         RETURN EOF;
      ELSE
         INC (inP);
         ch := inBuf[inP];
         RETURN Done;
      END;
   END Get;


   PROCEDURE Put (ch : CHAR);
   (* Ecrit un caractère dans un buffer fichier *)
      BEGIN
         INC (outP);
         outBuf[outP] := ch;
      END Put;


   PROCEDURE DoWrite (VAR f : File) : Status;
   (* Ecrit le buffer sur disque s'il est presque plein *)
      BEGIN
         IF outP < 400 THEN    (* il y a encore de la place *)
            RETURN Done;
         ELSE
            WriteNBytes (f, ADR (outBuf), outP);
            written := f.count;
            IF (written = outP) AND (f.res = done) THEN
               outP := 0;
               RETURN Done;
            ELSE
               RETURN Error;
            END;
         END;
      END DoWrite;

BEGIN (* initialisation du module *)
   ext := 0;
END Files.
```

(LA SUITE DE KERMIT DANS LE N° 11 DU MOIS DE MARS 1991)

10.2 : PROTECT

```
;=============================================================;
;    PROTECT VERSION 2.2                                      ;
;    Formatage de protection pour disquettes DOS              ;
;    Commandes d'assemblage :                                 ;
;                                                             ;
;                          MASM PROTECT                       ;
;                          LINK PROTECT                       ;
;                          EXE2BIN PROTECT PROTECT.COM        ;
;                                                             ;
;                                                             ;
;    (C) Alexei HOFFMAN pour DEBUG Magazine                   ;
;    Revu et corrigé par Bruno VASSEUR  (C) 1991              ;
;=============================================================;

cseg            segment 'CODE'
                assume cs:cseg,ds:cseg,es:cseg
                org 100h

;-------------
; Définitions des macros
;-------------

;---- msg strings ----

msg             macro string
                push ax
                push dx
                mov dx,offset string
                mov ah,09
                int 21h
                pop dx
                pop ax
endm

;---- cls - nettoyage écran ----

cls             macro
                push ax
                push bx
                push cx
                push dx
                xor al,al
                xor cx,cx
                mov dh,24
                mov dl,79
                mov bh,07
                mov ah,6
                int 10h
                xor bx,bx
                xor dx,dx
                mov ah,02
                int 10h
                pop dx
                pop cx
                pop bx
                pop ax
endm

;-----------
; Routine haute
;-----------

;---- codes divers ----
```

```
            cr equu 0dh
            lf equ 0ah

main:

;—— C'est parti ! ——

protect     proc near
              cls
              msg init_msg
              mov ah,01
              int 21h
              call change_base
              call reset_disk
              jc io_err

;—— Formatage piste 39 ——

              mov bx,offset format_table1
              call format
              jc io_err

;—— Ecriture secteur 112 ——

              mov sector,112
              call write_msg
              dec track
              mov bx,offset disk_base          ; pointe sur la nouvelle DPT
              mov byte ptr [bx+3],0            ; et en modifie l'octet 3.

;—— Formatage piste 38 ——

              mov bx,offset format_table2
              call format
              jc io_err

;—— Ecriture secteur 4 ——

              mov sector,4
              call write_msg

;—— Restauration des données originelles ——

              call restor_base
              call reset_disk
              jc io_err
            jmp fin

;—— Erreur E/S disque ——

io_err:
              msg msg7
              call restor_base

;—— Fin du programme ——

fin:
              msg msg6
            mov ax,4ch
              int 21h
protect     endp

;————————————
; Modification de la DPT
;————————————

change_base proc near
            push es
              mov ah,35h                       ; donne le segment
              mov al,1eh                       ; et l'offset de la
              int 21h                                  ; DPT d'origine.

;—— Sauvegarde de l'adresse de la DPT d'origine ——

              mov cs:save_off,bx               ; sauve l'offset et
              mov cs:save_seg,es               ; le segment de la DPT.

;—— Modification de l'adresse de la DPT ——

              mov bx,0078h                      ; place le segment et
              mov es:[bx],offset disk_base     ; l'offset de la
              mov es:[bx+2],cs                         ; nouvelle table
en
              pop es                                   ; 0000:0078.
              ret
change_base endp

;————————————————————
; Remise en ordre des paramètres disque
;————————————————————

restor_base proc near
            push es
              mov ax,0
              push ax
              pop es

;—— Restauration de l'adresse de la DPT d'origine ——

              mov bx,0078h
              mov ax,cs:save_off               ; restaure le segment
              mov es:[bx],ax                           ; et l'offset de
la
              mov ax,cs:save_seg               ; DPT originelle à
              mov es:[bx+2],ax                         ; l'adresse du
pointeur
              pop es                                   ; de
l'interruption 1Eh.
              ret
restor_base   endp

;————————————
; Formatage de la disquette
;————————————

format        proc near
              mov dl,drive
              mov dh,head
              mov ch,track
              mov cl,1
              mov al,9
              mov ah,05h
              int 13h
              ret
format        endp



;————————————————————
; Ecriture du message de protection (non-utilisé)
;————————————————————

write_msg   proc near
              mov ah,03
              mov al,01
              mov cl,sector
              mov ch,track
              mov dx,0
              mov bx,offset protect_msg
              int 13h
              ret
write_msg   endp

;————————————————————————
; Routine de Reset disque après modification de la DPT
;————————————————————————

reset_disk  proc near
              mov ah,00h
              int 13h
              ret
reset_disk  endp

;————————————————
; Ecran d'accueil et messages divers
;————————————————

init_msg    db cr,lf,lf,'PROTECT.COM - Verrouillage soft'
              db cr,lf,'(C) 1987  Alexeï HOFFMAN'
            db cr,lf,'Revu et corrigé par Bruno VASSEUR (C) 1991',cr,lf,lf
              db cr,lf,lf,'Mettez une disquette dans le '
              db 'lecteur A: ',cr,lf,'Démarrez le formatage en'
              db ' appuyant sur une touche.',cr,lf,lf,'$'
msg5          db cr,lf,lf,'Copie entièrement réussie.$'
msg6          db cr,lf,lf,'Fin de PROTECT.COM.',cr,lf,lf,'$'
msg7          db cr,lf,lf,'*** Erreur E/S disque ***$'

;————————————
; Données de formatage
;————————————

drive         db 0
track         db 39
head          db 0
sector        db 1

format_table1 db 027h,000h,070h,001h
              db 027h,000h,071h,001h
              db 027h,000h,072h,001h
              db 027h,000h,073h,001h
              db 027h,000h,074h,001h
              db 027h,000h,075h,001h
              db 027h,000h,076h,001h
              db 027h,000h,077h,001h
              db 027h,000h,078h,001h

format_table2 db 026h,000h,001h,000h
              db 026h,000h,002h,000h
              db 026h,000h,003h,000h
              db 026h,000h,004h,000h
              db 026h,000h,005h,000h
              db 026h,000h,006h,000h
              db 026h,000h,007h,000h
              db 026h,000h,008h,000h
              db 026h,000h,009h,000h

;————————————
; Message de protection (non-utilisé)
;————————————

protect_msg   db 'ATTENTION ! '
              db 'Disquette protégée par utilitaire anti-piratage   '
              db ' DEBUG Magazine               '
              db 'Auteurs :    Alexeï HOFFMAN et Bruno VASSEUR '
              db 72 dup ('P')

;————————————
```

```
; Données de la nouvelle DPT

;-------------------

disk_base       db 0cfh
                db 002h
                db 025h
                db 001h
                db 009h
                db 02ah
                db 0ffh
                db 050h
                db 0f6h
                db 020h
                db 004h

;-------------------------
; Vecteurs de sauvegarde de la DPT d'origine
;-------------------

save_off        dw ?
save_seg        dw ?
cseg            ends
                end main




;=============================================================;
;       VERIF.ASM                                             ;
;       Formattage de protection pour disquette DOS           ;
;       Routine de contrôle                                   ;
;       Commandes d'assemblage :                              ;
;                                                             ;
;                               MASM VERIF                    ;
;                               LINK VERIF                    ;
;                               EXE2BIN VERIF VERIF.COM       ;
;                                                             ;
;                                                             ;
;                                                             ;
;       (C) Alexeï HOFFMAN pour DEBUG Magazine                ;
;       Revu et corrigé par Bruno VASSEUR  (C) 1991           ;
;=============================================================;

;-------------
; Définition des macros
;-------------

cseg                    segment 'code'
                        assume cs:cseg,ds:cseg,es:cseg
                        org 100h

;---- msg string ----

msg                     macro string
                        push ax
                        push dx
                        mov dx,offset string
                        mov ah,09
                        int 21h
                        pop dx
                        pop ax
endm

;---- cls - nettoyage écran ----

cls                     macro
                        push ax
                        push bx
                        push cx
                        push dx
                        xor al,al
                        xor cx,cx
                        mov dh,24
                        mov dl,79
                        mov bh,07
                        mov ah,06
                        int 10h
                        xor bx,bx
                        xor dx,dx
                        mov ah,02
                        int 10h
                        pop dx
                        pop cx
                        pop bx
                        pop ax
endm

;-----------
; Routine haute
;-----------

;---- Codes divers ----

cr                      equ 0dh
lf                      equ 0ah
main:

;---- Et c'est parti ! ----

_test                   proc near
                        msg init_msg
                        mov ah,0

                        int 16h
                        call change_base
                        call reset_disk

;---- Routine de test proprement dite ----

retry:                  call restor_base
                        mov byte_sec,0
                        call change_base
                        mov sector,4
                        mov track,38
                        call read
                        jnc not_original
                        cmp ah,2
                        jne not_original
                        mov sector,9
                        call read
                        jnc not_original
                        cmp ah,4
                        jne not_original
                        call restor_base
                        mov byte_sec,1
                        call change_base
                        inc track
                        mov sector,70h
                        call read

                        jc not_original

;---- Test final ----

                        mov si,offset buffer
                        mov di,offset test_msg
                        mov cx,8
                        rep cmpsb
                        je not_original

;---- C'est une disquette originale ----

original: msg orig_msg
                        jmp fin

;---- C'est une infâme copie ! ----

not_original:           dec retry_count
                        cmp retry_count,0
                        je say_copy
                        jmp retry

say_copy: msg copy_msg
                        jmp fin

;---- Fin du programme ----

fin:                    call restor_base
                        call reset_disk
                        msg fin_msg
                        int 21h
_test                   endp

;-------------------------
; Modification de l'adresse de la DPT d'origine
;-------------------------

change_base     proc near
                push es
                  mov ah,35h                    ; donne le segment
                  mov al,1eh                    ; et l'offset de la
                  int 21h                               ; DPT d'origine.

;---- Sauvegarde de l'adresse de la DPT d'origine ----

                  mov cs:save_off,bx            ; sauve l'offset et
                  mov cs:save_seg,es            ; le segment de la DPT.

;---- Ecriture de l'adresse de la nouvelle DPT ----

                  mov bx,0078h                  ; place le segment et
                  mov es:[bx],offset disk_base  ; l'offset de la
                  mov es:[bx+2],cs                      ; nouvelle table
en
                  pop es                                ; 0000:0078.
                  ret
change_base     endp

;-------------------------
; Restauration de l'adresse de la DPT d'origine
;-------------------------

restor_base     proc near
                push es
                  mov ax,0
                  push ax
                  pop es
                  mov bx,0078h
                  mov ax,cs:save_off            ; restaure le segment
                  mov es:[bx],ax                        ; et l'offset de
la
                  mov ax,cs:save_seg            ; DPT originelle à
                  mov es:[bx+2],ax                      ; l'adresse du
pointeur
                  pop es                                ; de
l'interruption 1Eh.
```

```
                ret

restor_base     endp

;-------------------------
; Routine de lecture des secteurs à tester
;-------------------------

read            proc near
                  mov dl,drive
                  mov dh,head
                  mov ch,track
                  mov cl,sector
                  mov al,1
                  mov bx,offset buffer
                  mov ah,02h
                  int 13h
                  ret
read              endp

;-------------------------
; Reset disque après modification de la DPT
;-------------------------

reset_disk      proc near
                  mov ah,00h
                  int 13h
                  ret
reset_disk      endp

;-------------------
; Ecran d'accueil et messages divers
;-------------------


init_msg        db cr,lf,lf,'TEST DE PROTECTION',13,10
                  db cr,lf,' (C) 1987  Alexei HOFFMAN'
                db cr,lf,'Revu et corrigé par Bruno VASSEUR (C) 1991',cr,lf,lf
                  db cr,lf,lf,'Mettez une disquette dans le lecteur A: '
                  db 'et appuyez sur une touche ...$'
orig_msg        db cr,lf,'La disquette dans le lecteur A: '
                  db 'est protégée par PROTECT.$'
copy_msg        db cr,lf,'La disquette dans le lecteur A: '
                  db 'est une copie illicite !!!$'
fin_msg         db cr,lf,lf,'Fin de test.$'
test_msg        db 'PROTECT'

;---------------
; Données de formatage
;---------------

drive             db 0
track             db 38
head              db 0
sector            db 0

;---------------
; Données de la nouvelle DPT
;---------------

disk_base       db 0cfh
                  db 002h
                  db 025h
byte_sec        db 000h
                  db 009h
                  db 02ah
                  db 0ffh
                  db 050h
                  db 0f6h
                  db 020h
                  db 004h

;-------------------------
; Vecteurs de sauvegarde de la DPT originelle
;-------------------

save_off        dw ?
save_seg        dw ?
retry_count     db 3
buffer          equ this byte
cseg              ends
```

10.3 : Un Analyseur de Protocole de Transmission Série

```
                end main
{*********************************************}
{***                                       ***}
{***            Système de menus           ***}
{***    Pour l'analyseur de protocole série  ***}
{***                  par                  ***}
{***             Craig A. Lindley          ***}
{***                                       ***}
{***    Ver: 2.0  Dernière modif: 15/08/87   ***}
{***                                       ***}
{*********************************************}
```

```
CONST

 {Nombre max d'attributs vidéo - 1}
 AttribMax = 5;

 Attributs: ARRAY[0..AttribMax] OF
 RECORD
    f,              {Couleur des caractères}
    b: Integer;     {Couleur du fond}
 END

 {Les attributs sont:}
 {Normal, Surbrillance, Inverse}
 {Normal Clignotant, Surbrillance Clignotant, Inverse Clignotant}

  = ((f:7; b:0) , (f:15;b:0) , (f:0; b:7),
     (f:23;b:0) , (f:31;b:0) , (f:16;b:7));


 {Lignes d'affichage du menu par rapport à la ligne 1}
 LigneMenu1 = 2;
 LigneMenu2 = 3;
 LigneMenu3 = 4;
 LigneMenu4 = 5;


{dimensions max de l'A-arbre}
 LargeurNiveau2 = 7;
 LargeurNiveau3 = 5;
 LargeurNiveau4 = 7;

 home=      #199;    {code touche home}
 FlecheG=   #203;    {code touche curseur gauche}
 FlecheD=   #205;    {code touche curseur droit}
 toucheEnd= #207;    {code touche end}
 bs=        #08;     {code touche backspace}
 lf=        #10;     {code line feed}
 cr=        #13;     {code touche entrée}
 Esc=       #27;     {code touche escape}
 sp=        #32;     {code touche espace}

TYPE

 AttribType  = (Low ,High, Rev, LowBlink,
                HighBlink, RevBlink);

 {structure de données d'une entrée de menu de type A-arbre}
 {Voir l'article pour les détails}

 entree_menu=
 RECORD
    title:  STRING[10];
     desc:  STRING[40];
    chars:  STRING[8];
    index:  Byte;
    ccode:  Byte;
 END;

tree = ARRAY[0..LargeurNiveau2,
             0..LargeurNiveau3,
             0..LargeurNiveau4]
       OF entree_menu;

VAR

 ExitMenu,
 ExitProgram:       Boolean;

 ind1,ind2,
 ind3,
 selector,
 cmd_code,
 level:             Byte;

 {structure de données de type A-arbre pour sous menus}
 atree:             tree;


{******* Procédures Clavier et Affichage ********}

PROCEDURE Beep;

BEGIN

   Sound(1000);
   Pause(1);
   NoSound;

END;


FUNCTION GetKey : Byte;

VAR

 Ch: Char;
 B:  Byte;

 CurrentX,
 CurrentY:  Integer;

BEGIN

  {sauve position du curseur}
  {car on va yield}
```

```
  {et perdre le contrôle de l'affichage}

  CurrentX := WhereX;
  CurrentY := WhereY;

  {yield jusqu'à un appui sur une touche}

  WHILE NOT keypressed DO
      yield ;

  {restaure curseur}
  GoToXY(CurrentX,CurrentY);

  {lit la touche}
  read(kbd,Ch);

  {regarde si c'est un caractère d'échappement}
  IF (Ch = Esc) AND keypressed THEN
  BEGIN
    {Lit le deuxième caractère, et marque}
    {le caractère en modifiant son MSB}
    read(kbd,Ch);
    B := ord(Ch)+128;
  END
  ELSE
    {Sinon, B reçoit le code de la touche}

      B := ord(Ch);

  GetKey := B;

END;


FUNCTION repl (Count:Integer; ch:Char):FullString;

{duplique un caractère dans une chaine de}
{caractères de longueur spécifiée}

VAR

 i:               Integer;

BEGIN

   FOR i:=1 TO Count DO repl[i]:=ch;
   repl[0]:=Char(Count);

END;


PROCEDURE WriteString (Stng:FullString;
                       Attrib:AttribType);

BEGIN

   {Modifie couleur du fond et des caractères}
   {et écrit la chaine}

   WITH Attributs[ORD(Attrib)] DO
   BEGIN
      TextColor(f);
      TextBackGround(b);
   END;
   write(Stng);

   {mode vidéo normal}
   WITH Attributs[ORD(Low)] DO
   BEGIN
      TextColor(f);
      TextBackGround(b);
   END;

END;


PROCEDURE WriteStringAt (Stng:FullString;
                         Attrib:AttribType;
                         X,Y:Integer);

BEGIN

 GoToXY(X,Y);
 WriteString(Stng,Attrib);

END;


{********** Début des procédures de menu **********}

PROCEDURE center (Line,Width:Integer;
                  Outstr:FullString;
                  Attrib:AttribType);

{Centre une chaine et l'écrit avec ses attributs}
{à une ligne spécifiée, sur une longueur spécifiée}

BEGIN

   GoToXY(1,Line);
   Clreol;
   GoToXY((Width DIV 2)-(length(Outstr) DIV 2),
          Line);
   WriteString(Outstr,Attrib);

END;
```

```
PROCEDURE DrawFrame (x,y,width,height:Integer);

VAR

 top,bottom:      Str80;

BEGIN

   top:=   repl(width,Char(205));
   bottom:=repl(width,Char(205));
   WriteStringAt(top,High,x,y);
   WriteStringAt(bottom,High,x,y+height-1);

END;


PROCEDURE DrawMenuFrame;

BEGIN

   DrawFrame(1,1,80,6);
   center(LigneMenu1,78,
          ' Analyseur de Protocole Série - Menu ',Rev);


END;


PROCEDURE DisplayMenu (ind1,ind2,ind3,level,
                       selector:Byte);

VAR

 tind1,tind2,
 tind3,i:      Integer;
 attrib:       AttribType;
 titlestr:     FullString;

BEGIN
   titlestr:='— '+
     atree[ind1,ind2,ind3].title+' Selection'+' —';
   center(LigneMenu2,78,titlestr,Low);
   GoToXY(1,LigneMenu4);
   clreol;
   GoToXY(9,LigneMenu3);
   clreol;

   FOR i:= 1 TO atree[ind1,ind2,ind3].index DO
   BEGIN
      CASE level OF
         1: ind1:=i;
         2: ind2:=i;
         3: ind3:=i;
      END;
      IF selector <> i THEN  {entrée de menu non sélectionnée}
          attrib:=Low
      ELSE
      BEGIN
         attrib:=Rev;
         tind1:=ind1;
         tind2:=ind2;
         tind3:=ind3;
      END;
      WriteString(atree[ind1,ind2,ind3].title,
                  attrib);
      write('   '); {espaces pour separer les entrées}
   END;
   GoToXY(9,LigneMenu4);
   clreol;
   write(atree[tind1,tind2,tind3].desc);

END;


PROCEDURE ProcessCr (VAR ind1,ind2,ind3,level,
                     selector,cmd_code:Byte);

BEGIN
 {assigne l'élément sélectionné à l'index}
 CASE level OF
    1: ind1:=selector;
    2: ind2:=selector;
    3: ind3:=selector;
 END;
 {Si l'entrée est une entrée de menu terminale}
 IF atree[ind1,ind2,ind3].index = 0 THEN
 BEGIN
    {lit le code de commande associé}
    cmd_code:=atree[ind1,ind2,ind3].ccode;

    {exécute en fonction du niveau}
    CASE level OF
       1: ind1:=0;
       2: BEGIN
             ind1:=0;
             ind2:=0;
             ind3:=0;
             level:=1;
          END;
       3: BEGIN
             ind2:=0;
             ind3:=0;
             level:=2;
          END;
```

```
    END;

 END
 ELSE
 BEGIN
    {descend d'un niveau dans l'arborescence}
    level:=level+1;
 END;
 {sélectionne le 1er élément}
 selector:=1;

END;


PROCEDURE ProcessMenu (VAR ind1,ind2,ind3,level,
                       selector,cmd_code:Byte);

VAR

 key:          Char;
 position:     Integer;

BEGIN

   key:=upcase(GetKey);
   CASE key OF
    FlecheD: BEGIN

               selector:=selector+1;
               IF selector >
                  length(atree[ind1,ind2,ind3].chars) THEN
                   selector:=1;
             END;

    FlecheG: BEGIN
               selector:=selector-1;
               IF selector < 1 THEN
                  selector :=
                 length(atree[ind1,ind2,ind3].chars);
             END;

    toucheEnd: selector :=
              length(atree[ind1,ind2,ind3].chars);

      home: selector:=1;

        cr: ProcessCr(ind1,ind2,ind3,level,
                      selector,cmd_code);

       esc: cmd_code := 1;     {quitte}

'A'..'Z','0'..'9'  {1ère lettre d'une entrée de menu ?}
          : BEGIN
              position :=
                pos(key,atree[ind1,ind2,ind3].chars);
              IF position <> 0 THEN
              BEGIN              {c'est la 1ere lettre}
                 selector:=position;
                 ProcessCr(ind1,ind2,ind3,level,
                           selector,cmd_code);
              END
              ELSE      {ce n'est pas la 1ere lettre -> beep}
                 Beep;
            END;
      ELSE              {invalide -> beep}
          beep;
   END;

END;


{Procedure utilisée pour exécuter}
{la commande sélectionnée}
{Définie dans le programme principal}

PROCEDURE ProcessCmd (cmd_code: Byte); FORWARD;


FUNCTION DoMenu : Boolean;

VAR

 Line: Integer;

BEGIN

   {stoppe l'affichage des infos écran}
   UpdateScreenStatus := False;

   {efface les 6 lignes supérieures de l'écran}
   FOR Line := 1 TO 6 DO
   BEGIN
      GoToXY(1,Line);
      Clreol;
   END;

   {et l'avant dernière}
   GoToXY(1,24);
   Clreol;

   DrawMenuFrame;
   {efface le flag de sortie de menu}
   ExitMenu := False;

   {efface le flag de sortie de programme}
   ExitProgram := False;

   {sélectionne la 1ere entrée du menu par défaut}
   selector:=1;

   {code commande initialisé à 0 }
   cmd_code:=0;

   {1er niveau de hiérarchie}
   level:=1;

   {arborescence du menu racine}
   ind1:=0;
   ind2:=0;
   ind3:=0;

   REPEAT

      DisplayMenu (ind1,ind2,ind3,level,selector);
      ProcessMenu (ind1,ind2,ind3,level,selector,
                   cmd_code);
      ProcessCmd  (cmd_code);
      cmd_code:=0;  {reset code selected last}

   UNTIL ExitMenu;

   {autorise l'affichage des infos écran}
   UpdateScreenStatus := True;

   {retourne la condition de sortie du programme}
   DoMenu := ExitProgram;

END;


PROCEDURE Init_Menu;

{initialise l'arbre des menus}

BEGIN

   fillchar(atree,sizeof(atree),0);

   WITH atree[0,0,0] DO
   BEGIN
      title:='Menu';
      chars:='QPAIFCD';
      index:=7;
   END;

   WITH atree[1,0,0] DO
   BEGIN
      title:='Quitte';
      desc:='Quitte le menu';
      ccode:=1;
   END;

   WITH atree[2,0,0] DO
   BEGIN
      title:='Parametres';
      desc:='Modifie les paramètres série';
      chars:='QBSLP';
      index:=5;
   END;

   WITH atree[2,1,0] DO
   BEGIN
      title:='Quitte';
      desc:='Quitte ce sous-menu';
   END;

   WITH atree[2,2,0] DO
   BEGIN
      title:='Bauds';
      desc:='Modifie la vitesse de transmission';
      chars:='Q361249';
      index:=7;
   END;

   WITH atree[2,2,1] DO
   BEGIN
      title:='Quitte';
      desc:='Quitte ce sous-menu';
   END;

   WITH atree[2,2,2] DO
   BEGIN
      title:='300';
      ccode:=2;
   END;

   WITH atree[2,2,3] DO
   BEGIN
      title:='600';
      ccode:=3;
   END;

   WITH atree[2,2,4] DO
   BEGIN
      title:='1200';
      ccode:=4;
   END;

   WITH atree[2,2,5] DO
   BEGIN
      title:='2400';
```

```
    ccode:=5;

END;

WITH atree[2,2,6] DO
BEGIN
   title:='4800';
   ccode:=6;
END;

WITH atree[2,2,7] DO
BEGIN
   title:='9600';
   ccode:=7;
END;

WITH atree[2,3,0] DO
BEGIN
   title:='Stop Bits';
   desc:='Modifie le nombre de stop-bits';
   chars:='Q12';
   index:=3;
END;

WITH atree[2,3,1] DO
BEGIN
   title:='Quitte';
   desc:='Quitte ce sous-menu';
END;

WITH atree[2,3,2] DO
BEGIN
   title:='1';
   ccode:=8;
END;

WITH atree[2,3,3] DO
BEGIN
   title:='2';
   ccode:=9;
END;

WITH atree[2,4,0] DO
BEGIN
   title:='Lng Mot';
   desc:='Modifie le nombre de bits de données';
   chars:='Q5678';
   index:=5;
END;

WITH atree[2,4,1] DO
BEGIN
  title:='Quitte';
  desc:='Quitte ce sous-menu';
END;

WITH atree[2,4,2] DO
BEGIN
   title:='5';
   ccode:=10;
END;

WITH atree[2,4,3] DO
BEGIN
   title:='6';
   ccode:=11;
END;

WITH atree[2,4,4] DO
BEGIN
   title:='7';
   ccode:=12;
END;

WITH atree[2,4,5] DO
BEGIN
   title:='8';
   ccode:=13;
END;

WITH atree[2,5,0] DO
BEGIN
   title:='Parité';
   desc:='Modifie la parité';
   chars:='QIPS';
   index:=4;
END;

WITH atree[2,5,1] DO
BEGIN
   title:='Quitte';
   desc:='Quitte ce sous-menu';
END;

WITH atree[2,5,2] DO
BEGIN
   title:='Impaire';
   ccode:=14;
END;

WITH atree[2,5,3] DO
BEGIN
   title:='Paire';
   ccode:=15;
END;

WITH atree[2,5,4] DO

BEGIN
   title:='Sans';
   ccode:=16;
END;

WITH atree[3,0,0] DO
BEGIN
  title:='Affichage';
  desc:='Sélectionne les ports à afficher';
  chars:='Q12L';
  index:=4;
END;

WITH atree[3,1,0] DO
BEGIN
   title:='Quitte';
   desc:='Quitte ce sous-menu';
END;

WITH atree[3,2,0] DO
BEGIN
   title:='COM1';
   ccode:=17;
END;

WITH atree[3,3,0] DO
BEGIN
   title:='COM2';

   ccode:=18;
END;

WITH atree[3,4,0] DO
BEGIN
   title:='Les deux';
   ccode:=19;
END;

WITH atree[4,0,0] DO
BEGIN
   title:='Interrupt.';
   desc:='Conditions d'interruption';
   chars:='QCVMS';
   index:=5;
END;

WITH atree[4,1,0] DO
BEGIN
  title:='Quitte';
  desc:='Quitte ce sous-menu';
END;

WITH atree[4,2,0] DO
BEGIN
   title:='Canal';
   desc:='Spécifie un canal série';
   chars:='Q12';
   index:=3;
END;

WITH atree[4,2,1] DO
BEGIN
   title:='Quitte';
   desc:='Quitte ce sous-menu';
END;

WITH atree[4,2,2] DO
BEGIN
   title:='COM1';
   ccode:=20;
END;

WITH atree[4,2,3] DO
BEGIN
   title:='COM2';
   ccode:=21;
END;

WITH atree[4,3,0] DO
BEGIN
   title:='Valeur';
   desc:='Spécifie la valeur pour l'interruption';
   ccode:=22;
END;

WITH atree[4,4,0] DO
BEGIN
   title:='Mode';
   desc:='Sélectionne mode d'interruption';
   chars:='QJAV';
   index:=4;
END;

WITH atree[4,4,1] DO
BEGIN
   title:='Quitte';
   desc:='Quitte ce sous-menu';
END;

WITH atree[4,4,2] DO
BEGIN
   title:='Jusqu'à';
   desc:='Données affichée jusqu'à l'interruption';
   ccode:=23;
```

```
END;

WITH atree[4,4,3] DO
BEGIN
   title:='Après';
   desc:='Données affichées après l'interruption';
   ccode:=24;
END;

WITH atree[4,4,4] DO
BEGIN
   title:='Valide';
   desc:='Valide l'interruption';
   ccode:=25;
END;

WITH atree[4,5,0] DO
BEGIN
   title:='Stop';
   desc:='Stoppe l'attente d'interruption';
   ccode:=26;
END;

WITH atree[5,0,0] DO
BEGIN
   title:='Format';
   desc:='Format d'affichage';
   chars:='QAHE';
   index:=4;
END;

WITH atree[5,1,0] DO
BEGIN
   title:='Quitte';
   desc:='Quitte ce sous-menu';
END;

WITH atree[5,2,0] DO
BEGIN
   title:='Ascii';
   desc:='Affichage Ascii';
   chars:='QNP';
   index:=3;
END;

WITH atree[5,2,1] DO
BEGIN
   title:='Quitte';
   desc:='Quitte ce sous-menu';
END;

WITH atree[5,2,2] DO
BEGIN
   title:='Normal';
   desc:='Données seulement';
   ccode:=27;
END;

WITH atree[5,2,3] DO
BEGIN
   title:='Protocole';
   desc:='Données et protocole';
   ccode:=28;
END;

WITH atree[5,3,0] DO
BEGIN
   title:='Hexa';
   desc:='Affichage hexadécimal';
   chars:='QNP';
   index:=3;
END;

WITH atree[5,3,1] DO
BEGIN
   title:='Quitte';
   desc:='Quitte ce sous-menu';
END;

WITH atree[5,3,2] DO
BEGIN
   title:='Normal';
   desc:='Données seulement';
   ccode:=29;
END;

WITH atree[5,3,3] DO
BEGIN
   title:='Protocole';
   desc:='Données et protocole';
   ccode:=30;
END;

WITH atree[5,4,0] DO
BEGIN
   title:='Espaces';
   desc:='Insère des espaces dans l'affichage';
   ccode:=31;
END;

WITH atree[6,0,0] DO
BEGIN
   title:='Contrôle';
   desc:='Options de contrôle';
   chars:='QASEI';
   index:=5;
END;

WITH atree[6,1,0] DO
BEGIN
   title:='Quitte';
   desc:='Quitte ce sous-menu';
END;

WITH atree[6,2,0] DO
BEGIN
   title:='Acquisit.';
   desc:='Acquisition des données';
   ccode:=32;
END;

WITH atree[6,3,0] DO
BEGIN
   title:='Stop';
   desc:='Stoppe l'acquisition des données';
   ccode:=33;
END;

WITH atree[6,4,0] DO
BEGIN
   title:='Efface';
   desc:='Efface l'écran des données';
   ccode:=34;
END;

WITH atree[6,5,0] DO
BEGIN
   title:='Init.';
   desc:='Initialise l'analyseur';
   ccode:=35;
END;

WITH atree[7,0,0] DO
BEGIN
   title:='DOS';
   desc:='Quitte l'analyseur et retour au DOS';
   ccode:=36;
END;

END;



{$K-}              {Option de compilation-ne pas modifier}

{***********************************************}
{***                                         ***}
{***              Turbo Pascal               ***}
{***            Noyau multitâche             ***}
{***               écrit par                 ***}
{***            Craig A. Lindley             ***}
{***                                         ***}
{***    Ver: 2.0  Dernière modif: 15/08/87   ***}
{***                                         ***}
{***********************************************}

CONST

   task_stack_size  = 1000; {taille de la pile pour chaque tache}

   turbodseg: integer = 0; {stockage turbo pascal}
                           {valeur data segment}


TYPE

{états possible d'une tache}
   task_state = (ready,waiting,running);

{registres du 808X}
   register_type = RECORD
   CASE integer OF
      1: (ax,bx,cx,dx,bp,si,di,ds,es,flags:integer);
      2: (al,ah,bl,bh,cl,ch,dl,dh          :byte);
   END;


{structure de contrôle de tache}
{Task control block (tcb)}

 tcbptr = ^ tcb;        {pointeur sur tcb}

 tcb = RECORD
    link:  tcbptr;      {lien avec précédent tcb dans dseg}
    bptr:  integer;     {offset du pointeur de base dans sseg}
    state: task_state;  {ready, waiting, running}
    id:    byte;        {numéro de tache}
 END;

 waitptr = ^tcbptr;     {ptr sur ptr sur tcb}
                        {utilisé pour passer des}
                        {paramètres à la tache wait}

 semaphore = RECORD     {type Semaphore}
    count:  integer;    {nombre de signaux}
    signal: tcbptr;     {pointeur sur la tache wait}
                        {s'il en existe une}
 END;
```

```
{******** Variables du multitache *********}

VAR

   cp,                    {pointeur tache courante}
   new_tcb_ptr,           {pointeur nouveau tcb dans dseg}
   temp_ptr:   tcbptr;

   waitfor:    waitptr; {adresse de l'item à attendre}
   stk,bp:     integer; {variables pour modifier les}
                        {registres sp et bp du 808X}
   frame_ptr:  integer; {pointeur de pile}
   next_id:    integer; {numéro de la tache suivante}
   i:          integer;
   child_process: boolean; {flag duplication de tache réussie}


{******** Procédures Multitaches ********}


PROCEDURE Fork;         {duplique une nouvelle tache}

{Cette procedure manipule la pile de Turbo Pascal}
{et la répartit entre les différentes tâches}

BEGIN

   child_process:=false;   {indique la tache parent}
                           {jusqu'à ce qu'il soit prouvé}
                           {le contraire}

   {teste s'il reste de la place pour une nouvelle tache}

   IF abs(frame_ptr - task_stack_size) > 0 THEN
   BEGIN                   {il en reste}
      INLINE($89/$26/stk); {lit le registre Sp}
                           {pour calculer le pointeur Bp}
      cp^.bptr:=stk+2;     {sauve Bp}
      new(new_tcb_ptr);    {alloue nouveau tcb}

      {relie le nouveau tcb dans la boucle du scheduler}
      {active la tache et attribue un ID}

      new_tcb_ptr^.link:=cp^.link;
      cp^.link:=new_tcb_ptr;
      new_tcb_ptr^.state:=running;
      next_id:=next_id+1;
      new_tcb_ptr^.id:=next_id;

      cp^.state:=ready;    {ancienne tache devient "prêt"}

      {recopie ancienne pile dans la nouvelle}
      FOR i:=0 TO 5 DO
         mem[sseg:frame_ptr-6+i]:=mem[sseg:stk+i];

      {Bp pointe sur la nouvelle pile}

      memw[sseg:frame_ptr-4]:=frame_ptr;
      bp:=frame_ptr-4;

      INLINE($8B/$2E/bp);  {Affecte au registre Bp 808X}
                           {cette nouvelle valeur}

     {reserve de l'espace pour la pile}
     frame_ptr:=frame_ptr-task_stack_size;
     cp:=new_tcb_ptr;      {cp pointe vers la nouvelle tache}
     child_process:=true;  {indique processus fils}
   END;

END;


(*
Cette procédure en Pascal est en commentaire.
La version en assembleur suit.

PROCEDURE Yield;

{Cette procédure fait rendre la main de la tache}
{en cours au CPU, et active la prochaine}
{tache prête}

BEGIN

   child_process:=false;   {initialise variable}
   IF cp^.link <> cp THEN  {il doit y avoir plus}
                           {d'une tache dupliquée pour}
                           {pouvoir changer de tache}
   BEGIN
      INLINE($89/$26/stk); {lit registre 808X sp}
      cp^.bptr:=stk+2;     {sauve le pointeur Bp dans}
                           {la structure de tache courante}
      cp^.state:=ready;    {la tache à interrompre devient "prête"}
      temp_ptr:=cp;

      {recherche la prochaine tache "prete" dans}
      {la boucle du scheduler. Il doit y en}
      {avoir au moins une}

      WHILE (temp_ptr^.link^.state <> ready) DO
         temp_ptr:=temp_ptr^.link;

      cp:=temp_ptr^.link;  {cp pointe sur la nouvelle tache}
      cp^.state:=running;  {indique tache active}
      bp:=cp^.bptr;        {lit pointeur bp de la tache}
      INLINE($8B/$2E/bp);  {le restaure dans le registre 808X bp}

   END
   ELSE
   BEGIN
      writeln('Impossible de changer de tache, une seule tache est en cours');
      halt;
   END;

END;
*)

PROCEDURE Yield;

{cette version en assembleur augmente}
{la vitesse. Voir la version en Pascal}
{pour les commentaires}

BEGIN

   IF cp^.link <> cp THEN  {il doit y avoir plus}
                           {d'une tache dupliquée pour}
                           {pouvoir changer de tache}
   BEGIN
      INLINE($C6/$06/child_process/$00/
                           {child_process reçoit "false"}
             $C4/$3E/cp/   {les di,[cp]}
             $89/$E0/      {mov ax,sp}
             $05/$02/$00/  {add ax,2}
             $26/          {es:}
             $89/$45/$04/  {mov [di+4],ax}

             $26/          {es:}
             $C6/$45/$06/$00/ {mov byte ptr [di+6],0}
             $89/$FB/      {L1: mov bx,di}
             $26/          {es:}
             $C4/$1F/      {les bx,[bx]}
             $26/          {es:}
             $80/$7F/$06/$00/ {cmp byte ptr [bx+6],0}
             $74/$04/      {je L2}
             $89/$DF/      {mov di,bx}
             $EB/$F0/      {jmp L1}
             $89/$1E/cp/   {L2: mov [cp],bx}
             $8C/$06/cp+2/ {    mov [cp+2],es}
             $26/          {es:}
             $C6/$47/$06/$02/ {mov byte ptr [bx+6],2}
             $26/          {es:}
             $8B/$6F/$04); {mov bp,[bx+4]}
   END
   ELSE
   BEGIN
      writeln('Impossible de changer de tache, une seule tache est en cours');
      halt;
   END;

END;

(*
PROCEDURE Wait;  {met la tache courante en état d'attente}
                 {jusqu'à ce qu'un Send soit envoyé}

{Dû à des contraintes de ce noyau, aucun paramètre}
{ne peut être passé directement à la procédure Wait.}
{Pour pallier à cela, une variable globale}
{appelée waitfor est utilisée. L'adresse du tcbptr}
{qui représente la tache à faire attendre doit}
{être stockée dans waitfor.}
{Voir les routines Fifo pour un exemple}
{d'utilisation de Wait.}

BEGIN

   child_process:=false;
   IF cp^.link <> cp THEN  {il doit y avoir plus}
                           {d'une tache dupliquée pour}
                           {pouvoir changer de tache}
   BEGIN
      waitfor^ := cp;      {waitfor pointe sur la}
                           {tache courante}

      INLINE($89/$26/stk); {lit le registre 808X sp}
      cp^.bptr:=stk+2;     {le sauve dans le tcb courant}
      cp^.state:=waiting;  {la tache attend}
      temp_ptr:=cp;

      {recherche la prochaine tache "prete" dans}
      {la boucle du scheduler. Il doit y en}
      {avoir au moins une}

      WHILE (temp_ptr^.link^.state <> ready) DO
         temp_ptr:=temp_ptr^.link;

      cp:=temp_ptr^.link;  {cp pointe sur la nouvelle tache}
      cp^.state:=running;  {indique tache active}
      bp:=cp^.bptr;        {lit pointeur bp de la tache}
     INLINE($8B/$2E/bp);   {le restaure dans le registre 808X bp}

   END
   ELSE
   BEGIN
      writeln('Impossible de bloquer la seule tache active');
      halt;
   END;

END;
*)
```

```
PROCEDURE Wait;  {met la tache courante en état d'attente}
                 {jusqu'à ce qu'un Send soit envoyé}

BEGIN

   IF cp^.link <> cp THEN  {il doit y avoir plus}
                           {d'une tache dupliquée pour}
                           {pouvoir changer de tache}
   BEGIN
      waitfor^ := cp;      {waitfor pointe sur la}
                           {tache courante}
      INLINE($C6/$06/child_process/$00/
             $C4/$3E/cp/       {les di,[cp]}
             $89/$E0/          {mov ax,sp}
             $05/$02/$00/      {add ax,2}
             $26/              {es:}
             $89/$45/$04/      {mov [di+4],ax}
             $26/              {es:}
             $C6/$45/$06/$01/  {mov byte ptr [di+6],1}
             $89/$FB/          {L1: mov bx,di}
             $26/              {es:}
             $C4/$1F/          {les bx,[bx]}
             $26/              {es:}
             $80/$7F/$06/$00/  {cmp byte ptr [bx+6],0}
             $74/$04/          {je L2}
             $89/$DF/          {mov di,bx}
             $EB/$F0/          {jmp L1}
             $89/$1E/cp/       {L2: mov [cp],bx}
             $8C/$06/cp+2/     {    mov [cp+2],es}
             $26/              {es:}

             $C6/$47/$06/$02/  {mov byte ptr [bx+6],2}
             $26/              {es:}
             $8B/$6F/$04);     {mov bp,[bx+4]}

   END
   ELSE
   BEGIN
      writeln('Impossible de bloquer la seule tache active');
      halt;
   END;

END;


PROCEDURE Send(VAR s:tcbptr);

{Supprime l'état d'attente d'une tâche}

BEGIN

   s^.state:=ready;     {l'état de la tache est "prêt"}
   s:=NIL;              {efface pointeur}

END;



PROCEDURE Pause(t:integer);

{Pause l'exécution d'une tache de}
{"t" quarts de secondes}

 FUNCTION tic_count : integer;

 {Lit la valeur courante du timer bios}

 VAR

    regs: register_type;

 BEGIN

    regs.ax:=0;          {lecture timer}
    intr($1A,regs);
    tic_count:=regs.dx;  {LSB du compteur dans dx}

 END;


VAR

   tics,i: integer;

BEGIN

   tics:=0;             {Valeur initiale du compteur à zéro}
   IF t > 0 THEN        {si c'est une valeur correcte}
   BEGIN
      FOR i:=1 TO t DO  {250 msec = 4.55 tops}
         IF odd(i) THEN {cet algorithme permet}
                        {une approximation :}
            tics:=tics+4 {250 msec = 4.5 tops}
         ELSE
            tics:=tics+5;
      {ajoute le nb de tops à attendre à la valeur}
      {du timer pour obtenir l'instant de fin}
      tics:=tics+tic_count;

      REPEAT
         yield;     {boucle jusqu'à ce que le}
                    {temps soit écoulé}
      UNTIL tics <= tic_count;
   END
   ELSE

      writeln('Mauvais nombre de périodes');

END;


PROCEDURE Init_Kernel;

{Initialise le multitache}
{Modifie le TCB de la tache 0}
{et indique que c'est la tache active}

BEGIN

   turbodseg:=dseg;      {sauve data segment de turbo pascal}
   frame_ptr:= $FFFE;    {pointeur de pile initial}
   next_id:=0;           {ID de la première tache}
   new(new_tcb_ptr);     {nouveau tcb dans dseg}
   cp:=new_tcb_ptr;      {cp pointe sur le tcb}
   cp^.link:=cp;         {pointe sur lui-même}
   cp^.state:=running;   {état actif}
   cp^.id:=next_id;      {id = 0}

   frame_ptr:=frame_ptr-task_stack_size;

END;



{********* Procédures de Semaphore **********}

PROCEDURE Initialize_semaphore(VAR s:semaphore);

{Initialise une structure de données semaphore}

BEGIN

   s.count := 0;        {indique que la ressource}
                        {est libre}
   s.signal:=NIL;       {et que personne}
                        {ne l'attend}

END;


PROCEDURE Alloc(VAR s:semaphore);

{Cette procédure alloue l'usage exclusif}
{d'une ressource à une tache qui l'exécute}
{Cette demande est maintenue meme quand la}
{tache rend la main au CPU par un}
{appel à la fonction yield}

BEGIN

   WHILE s.count <> 0 DO  {attend qu'une ressource}
                          {controlée par un sémaphore}
                          {devienne disponible}
   BEGIN
      waitfor := addr (s.signal);
      wait;
   END;                   {alors}
   s.count:=1;            {la réserve}

END;


PROCEDURE Dealloc(VAR s:semaphore);

{Cette procédure désalloue une ressource.}
{Notez que cette routine appelle yield,}
{etque la ressource désallouée peut}
{être utilisée immédiatement}

BEGIN

   s.count:=0;      {marque la ressource libre}
   send(s.signal); {et réveille la tache qui attend}
   yield;           {donne une chance aux autres taches}

END;

{Fin des procédure du Noyau Multitache}



{***********************************************}
{***                                         ***}
{***            Procédures RS 232            ***}
{***    Pour l'analyseur de protocole série  ***}
{***                    par                  ***}
{***              Craig A. Lindley           ***}
{***                                         ***}
{***    Ver: 2.0  Dernière modif: 15/08/87   ***}
{***                                         ***}
{***********************************************}

CONST

 COM1 = $3f8;              {adresse port COM1}
 COM2 = $2f8;              {adresse port COM2}

 {table des vitesses en bauds supportées}
 {par le 8250. Varie de 50 à 9600}
```

```
 BaudRate: ARRAY[1..15] OF Integer =

   ($900,$600,$417,$359,$300,$180,$0C0,$060,
    $040,$03A,$030,$020,$018,$010,$00C);

 {registres 8259}

 Int_Mask_Reg = $21; {registre de masque d'interruption}
 Int_Cmd_Reg  = $20; {registre de commande}
 End_Int_Cmd  = $20; {fin de commande d'interruption}

 {offsets par rapport à PortAddr pour}
 {les registres du 8250}

 DLL            = 0;
 DLM            = 1;
 Int_Enable_Reg = 1;
 Int_Id_Reg     = 2;
 LineControl    = 3;
 ModemControl   = 4;
 LineStatus     = 5;
 ModemStatus    = 6;

 {Définitions des bits de Status}

 DataRdyBit     = $01;
 DTRBit         = $01;
 Out2Bit        = $08;
 ORBit          = $01;
 RTSBit         = $02;
 PEBit          = $02;
 FEBit          = $04;
 BrkBit         = $08;
 CTSBit         = $10;

 DSRBit         = $20;
 TxRdyBit       = $20;
 RIBit          = $40;
 CDBit          = $80;

TYPE

 ParityType = (odd,even,none);

VAR

 {Variables globales des paramètres du port COM}
 {Utilisées par la fonction SetNewCOMParameter}

 COM_Rate,
 COM_StopBits,
 COM_DataBits:  Integer;
 COM_Parity:    ParityType;


{************** Procedures Série ***************}

PROCEDURE Enable_Serial_Device (PortAddr:Integer);

VAR

 Temp:   Byte;

BEGIN

   {efface des données qui pourraient être encore}
   {dans le 8250 en lisant le port de données,}
   {le registre d'identification d'interruption,}
   {et le registre de status, et le registre}
   {de status du modem.}

    Temp := port[PortAddr];
    Temp := port[PortAddr+Int_Id_Reg];
    Temp := port[PortAddr+LineStatus];
    Temp := port[PortAddr+ModemStatus];

    {lit le registre de masque d'interruption}
    {du 8259, et met les bits correspondant à}
    {IRQ3 et IRQ4 à l'état bas pour autoriser}
    {les instructions correspondantes.}
    {Ecrit le résultat dans le 8259.}

    Temp := port[Int_Mask_Reg];
    IF PortAddr = COM1 THEN
       Temp := Temp AND $EF
    ELSE
       Temp := Temp AND $F7;
    port[Int_Mask_Reg] := Temp;

    {Out2, DTR et RTS du 8250 à l'état haut}
    port[PortAddr+ModemControl] :=
         Out2Bit + DTRBit + RTSBit;

    {autorise toutes les interruptions}
    {sauf Tx}
    port[PortAddr+Int_Enable_Reg] := $0D;
    delay(100);

END;


PROCEDURE Disable_Serial_Devices;

VAR

 Temp:   Byte;

BEGIN

    {Lit le masque d'interruption du 8250,}
    {positionne les bits correspondant à}
    {IRQ3 et IRQ4 à l'état haut pour interdire}
    {ces interruptions.}
    {Positionne Out2 à l'état bas pour COM1}
    {et COM2 pour interdire toute interruption}
    {de la carte asynchrone, et finalement}
    {interdit toutes les interruptions}

    Temp := port[Int_Mask_Reg];
    Temp := Temp OR $18;
    port[Int_Mask_Reg] := Temp;
    port[COM1+ModemControl] := 0;
    port[COM2+ModemControl] := 0;
    port[COM1+Int_Enable_Reg] := 0;
    port[COM2+Int_Enable_Reg] := 0;

END;


PROCEDURE COM1_ISR;

{routine de service d'interruption COM1}

VAR

 SerialInfo:  DataRec;

 Temp:        Integer;

 LineState,

   ModemState:  Byte;

BEGIN

    {Lit les registres d'état de la ligne}
    {et du modem, et les combine pour}
    {former COM1_Status.}

    LineState := port[COM1+LineStatus];
    ModemState:= port[COM1+ModemStatus];
    Temp := (LineState SHR 1) AND $0F;

    {Les 8 bits inférieurs de COM1_Status}
    {contiennent l'état de l'UART}

    COM1_Status := (ModemState AND $F0) OR lo(Temp);

    {s'il y a des données à recevoir,}
    {et que l'on lit des données}

    IF (((LineState AND DataRdyBit) <> 0) AND
        COM1_Data_Acquire) THEN
    BEGIN
       {s'il n'y a plus de place dans}
       {la pile fifo en entrée de COM1}
       IF COM1_Input_Fifo.Ovd.Count <
                  SerialDataFifoSize THEN
       BEGIN
          {Ecrit la donnée reçue et le status}
          {dans la pile fifo}
          SerialInfo.Data   := port[COM1];
          SerialInfo.Status := COM1_Status;
          PutSerialData(SerialInfo,COM1_Input_Fifo);
       END
       ELSE
       BEGIN
          writeln('Pile fifo d'entrée COM1 pleine');
          halt;
       END;
    END;

    {signale fin d'interruption au 8259}
    port[Int_Cmd_Reg] := End_Int_Cmd;

END;


PROCEDURE COM2_ISR;

{routine de service d'interruption COM2}

VAR

 SerialInfo:  DataRec;

 Temp:        Integer;

 LineState,
 ModemState:  Byte;

BEGIN

    {Lit les registres d'état de la ligne}
    {et du modem, et les combine pour}
    {former COM2_Status.}

    LineState := port[COM2+LineStatus];
    ModemState:= port[COM2+ModemStatus];
    Temp := (LineState SHR 1) AND $0F;

    {Les 8 bits inférieurs de COM2_Status}
```

```
  {contiennent l'état de l'UART}

  COM2_Status := (ModemState AND $F0) OR lo(Temp);

  {s'il y a des données à recevoir,}
  {et que l'on lit des données}

  IF (((LineState AND DataRdyBit) <> 0) AND
      COM2_Data_Acquire) THEN
  BEGIN
     {s'il n'y a plus de place dans}
     {la pile fifo en entrée de COM1}
     IF COM2_Input_Fifo.Ovd.Count <
                  SerialDataFifoSize THEN
     BEGIN
        {Ecrit la donnée reçue et le status}
        {dans la pile fifo}
        SerialInfo.Data   := port[COM2];
        SerialInfo.Status := COM2_Status;
        PutSerialData(SerialInfo,COM2_Input_Fifo);
     END
     ELSE
     BEGIN
        writeln('Pile fifo d'entrée COM2 pleine');
        halt;
     END;
  END;

  {signale fin d'interruption au 8259}
  port[Int_Cmd_Reg] := End_Int_Cmd;

END;


PROCEDURE COM1_Int_Service_Routine;

BEGIN

 INLINE($50/$53/$51/$52/$57/    {Push ax,bx,cx,dx,}
        $56/$06/$1e/            {di,si,es,ds}
        $2e/$a1/turbodseg/      {mov ax,cs:turbodseg}
        $8e/$d8/                {mov ds,ax}
        $fb);                   {sti}

 COM1_ISR;

 {Fin d'interruption standard}

 INLINE($fa/$1f/$07/$5e/$5f/    {supprime interruptions}
        $5a/$59/$5b/$58/        {Pop ds,es,si,di,}
                                {dx,cx,bx,ax}
        $5d/$5d/$cf);           {jette sp, restaure}
                                {Bp et iret}

END;


PROCEDURE COM2_Int_Service_Routine;

BEGIN

 INLINE($50/$53/$51/$52/$57/    {Push ax,bx,cx,dx,}
        $56/$06/$1e/            {di,si,es,ds}
        $2e/$a1/turbodseg/      {mov ax,cs:turbodseg}
        $8e/$d8/                {mov ds,ax}
        $fb);                   {sti}

 COM2_ISR;

 {Fin d'interruption standard}

 INLINE($fa/$1f/$07/$5e/$5f/    {supprime interruptions}
        $5a/$59/$5b/$58/        {Pop ds,es,si,di,}
                                {dx,cx,bx,ax}
        $5d/$5d/$cf);           {jette sp, restaure}
                                {Bp et iret}

END;


PROCEDURE Install_Serial_Handlers;

BEGIN

 WITH regs DO       {installe pour IRQ3 & 4}
 BEGIN
    ah := $35;        {Lit vecteurs d'interruption}
    al := $0B;        {pour IRQ3}
    msdos(regs);      {appelle dos pour lire le vecteur}
    OldIRQ3_CS := es;{sauve code segment}
    OldIRQ3_IP := bx;{et vecteur d'interruption}

    ah := $35;        {Lit vecteurs d'interruption}
    al := $0C;        {pour IRQ4}
    msdos(regs);      {appelle dos pour lire le vecteur}
    OldIRQ4_CS := es;{sauve code segment}
    OldIRQ4_IP := bx;{et vecteur d'interruption}

    ah := $25;        {modifie vect. d'int.}
    al := $0C;        {pour IRQ4}
    ds := cseg;
    dx := ofs(COM1_Int_Service_Routine);
    msdos(regs);

    ah := $25;        {modifie vect. d'int.}
    al := $0B;        {pour IRQ3}
    ds := cseg;
    dx := ofs(COM2_Int_Service_Routine);
    msdos(regs);

 END;
END;


PROCEDURE Remove_Serial_Handlers;

BEGIN
   {Restaure les vecteurs d'interruption}
   {de IRQ3 et IRQ4}
   WITH regs DO
   BEGIN
      ah := $25;
      al := $0C;
      ds := OldIRQ4_CS;
      dx := OldIRQ4_IP;
      msdos(regs);

      ah := $25;
      al := $0B;
      ds := OldIRQ3_CS;
      dx := OldIRQ3_IP;
      msdos(regs);
   END;

END;


PROCEDURE Set_Serial_Parameters
     (PortAddr,Baud,StopBits,DataBits:Integer;
      Parity: ParityType);
VAR

 Temp,
 Rate:   Integer;

BEGIN

   {positione DLAB à un}
   {pour accéder aux registres de vitesse}
   port[PortAddr+LineControl] := $80;

   {lit la vitesse dans la table}
   Rate := BaudRate[Baud];

   {MSB dans registre fort}
   port[PortAddr+DLM] := hi(Rate);

   {LSB dans registre faible}
   port[PortAddr+DLL] := lo(Rate);

   {le nb de bits de données sont rangés dans}
   {les bits 0 et 1, le nb de stop bits dans}
   {le bit 2, et la parité dans les bits 3 et 4}

   Temp := (DataBits - 5) AND $03;
   Temp := Temp OR ((StopBits - 1) SHL 2);
   CASE Parity OF
     odd:  Temp := Temp + $08;
    even: Temp := Temp + $18;
    none: ;
   END;

   {Met le bit DLAB à zéro}
   {et écris les modifications des paramètres}
   port[PortAddr+LineControl] := lo(Temp);
   delay(100);

END;


FUNCTION Get_Serial_Status (PortAddr:Integer)
              :Integer;

VAR

 Temp:  Integer;

BEGIN

   {Lit le status sur 16 bits du port COM}
   {groupé de la manière suivante}
   {ModemStatus:LineStatus}

   Temp := port[PortAddr+ModemStatus];
   Temp := Temp SHL 8;
   Temp := Temp OR port[PortAddr+LineStatus];
   Get_Serial_Status := Temp;

END;


PROCEDURE SetNewCOMParameter;

BEGIN

   {Desactive les ports COM}
   Disable_Serial_Devices;
```

```
   {Initialise les ports COM}

   {avec les paramètres globaux}

   Set_Serial_Parameters(COM1,COM_Rate,
         COM_StopBits,COM_DataBits,COM_Parity);
   Set_Serial_Parameters(COM2,COM_Rate,
         COM_StopBits,COM_DataBits,COM_Parity);

   {Active les ports COM}
   Enable_Serial_Device(COM1);
   Enable_Serial_Device(COM2);

END;

{***********************************************}
{***        Analyseur de protocole série      ***}
{***           écrit en Turbo Pascal          ***}
{***                    par                   ***}
{***              Craig A. Lindley            ***}
{***                                          ***}
{***    Ver: 2.0  Dernière modif: 15/08/87    ***}
{***                                          ***}
{***********************************************}

{$K-,U-,C-,G30,D-}

{ -- Notes sur les options de compilation -- }
{ K-    Pas de test de pile sinon le multitache }
{       ne fonctionne pas.                      }
{ U-.C- Ne test pas la touche break pour        }
{       accélérer l'affichage écran.            }
{ G30,D- Bufferise l'entrée standard (clavier)  }
{       et supprime les tests des périphériques.}
{       Le clavier est plus rapide et son entrée}
{       est bufferisée.                         }

CONST

  HexDigits:  STRING[16] = '0123456789ABCDEF';

  AsciiStrs: ARRAY[0..31] OF STRING[3] =
    ('Nul', 'Soh' , 'Stx','Etx' ,'Eot' , 'Enq' , 'Ack', 'Bel' ,
    'Bs' ,'Ht' ,'Lf' ,'Vt' ,'Ff' ,'Cr' ,'So' ,'Si' ,
    'Dle', 'Dc1' , 'Dc2' , 'Dc3' , 'Dc4' , 'Nak' , 'Syn' , 'Etb' ,
    'Can', 'Em' ,'Sub','Esc','Fs' ,'Gs' ,'Rs' ,'Vs');


  SerialDataFifoSize = 2000;  {Taille d'une pile de données fifo}
  DisplayFifoSize    = 3000;  {Taille d'une pile d'affichage fifo}

  {$I multi.pas}                          {inclus le}
                                          {noyau multitache}

TYPE

  FullString     = STRING [255];
  Str80          = STRING [80];
  Str3           = STRING[3];

  DataRec = RECORD
    Data,
    Status:  Byte;
  END;

  DisplayRec = RECORD
    Tag: (FromCOM1,FromCOM2);
    DR:  DataRec;
  END;

{Cette structure d'en-tête de pile fifo est}
{la même pour tous les types de pile fifo,}
{quel que soit le type d'objet stocké dans}
{la pile. Deux types de pile fifo sont}
{définis plus loin.}

  OverHead = RECORD       {structure d'en-tête}
                          {de pile  fifo}
    Count,                {nb d'éléments ds la pile}
    Inptr,                {pointeur vers l'endroit où}
                          {sont stockés les données}
    Outptr:   Integer;    {pointeur vers l'endroit où}
                          {sont acquises les données}
    NotEmpty,             {pointeurs vers les tâches}
    NotFull:  tcbptr;     {qui attendent les données}
END;


{définition d'une pile fifo de données série}
SerialDataFifo = RECORD
  Ovd:  OverHead;               {entete fifo}
  Data: ARRAY[1..SerialDataFifoSize]
        OF DataRec;             {zone de données}
END;

{définition d'une pile fifo d'affichage écran}
DisplayFifoType = RECORD
  Ovd:  OverHead;
  Data: ARRAY[1..DisplayFifoSize]
        OF DisplayRec;
END;

DisplayTriggerType = (Before,After);

VAR
  regs:  register_type;

{stockage du code segment et de l'adresse}
{du vecteur des interruptions}

 {originales IRQ3 et IRQ4}

 OldIRQ3_CS,
 OldIRQ3_IP,
 OldIRQ4_CS,
 OldIRQ4_IP:  Integer;

{stockage du status de l'UART}

 OldCOM1_Status,
 OldCOM2_Status,
 COM1_Status,
 COM2_Status: Byte;

{Flags d'affichage formaté binaires}

 UpdateScreenStatus,
 AsciiDisplay,
 HandShakeDisplay,
 AddSpace,

{flags d'affichage des données}
{vrai si les données du port COM spécifié}
{sont marquées et doivent être recopiées}
{dans la pile fifo d'affichage}

 COM1_Display_Data,
 COM2_Display_Data,

{flag booléens d'acquisition de données}
{Contrôle l'acquisition des données}
{par les routines de lecture du port}
{série par interruptions.}
{vrai si les données du port sont}
{lues par les interruptions}

 COM1_Data_Acquire,
 COM2_Data_Acquire,

{Variables utilisées pour les fonctions de point d'arret}

 TriggerEnabled,
 COM1_Is_Triggered,
 COM2_Is_Triggered:     Boolean;
 TriggerPattern:        Integer;
 TriggerMode:           DisplayTriggerType;

 {déclarations des piles Fifo}

 COM1_Input_Fifo,
 COM2_Input_Fifo,
 COM1_Output_Fifo,
 COM2_Output_Fifo:      SerialDataFifo;

 DisplayFifo:    DisplayFifoType;

{Chaines de caractère pour l'affichage}
{formaté de l'écran, utilisées lors de l'exécution.}

 Line1, Line2,
 Line3, Line4,
 Line5, Line6,
 Line24:        Str80;

{Stockage de la position du curseur}
{dans la procédure DisplayCOMData}

 OldXPos,
 OldYPos:  Integer;

 {Sémaphore pour le contrôle de l'écran}
 ScreenAccess: Semaphore;

{************** Procedures de gestion de pile FIFO ************}

PROCEDURE Init_Fifos;

  PROCEDURE Initialize_fifo(VAR o:OverHead);

 {Initialise l'en-tête d'une pile FIFO.}
 {Cette procédure fonctionne avec n'importe quel}
 {type de pile Fifo. La pile apparait vide.}

 BEGIN

   o.Count  := 0;          {count est vide}
   o.Inptr  := 1;          {pointeur sur lere entrée}
   o.Outptr := 1;          {lit et écrit dans la}
                           {1ère entrée}
   o.NotEmpty :=NIL;       {pas de signaux}
   o.NotFull  :=NIL;

 END;

 BEGIN

   Initialize_fifo(COM1_Input_fifo.Ovd);
   Initialize_fifo(COM1_Output_fifo.Ovd);
   Initialize_fifo(COM2_Input_fifo.Ovd);
   Initialize_fifo(COM2_Output_fifo.Ovd);
   Initialize_fifo(DisplayFifo.Ovd);
```

```
END;

PROCEDURE PutSerialData (d:DataRec;
                         VAR f:SerialDataFifo);

BEGIN

  WITH f.Ovd DO
  BEGIN                              {teste si fifo pleine}
     IF Count = SerialDataFifoSize THEN
     BEGIN                           {si oui, attend}
        waitfor := addr (NotFull);
        wait;

     END;                            {quand non pleine,}
     Count:=Count+1;                 {incrémente count}
     f.data[Inptr] :=d;              {stocke la donnée}
     Inptr:=Inptr+1;                 {incrémente pointeur d'entrée}
     IF Inptr > SerialDataFifoSize THEN
        Inptr:=1;                    {"enroule" ptr si nécéssaire}

     {si une tache attend après cette fifo}
     {elle est réveillée}
     IF NotEmpty <> NIL THEN
                      send(NotEmpty);
  END;

END;

PROCEDURE GetSerialData(VAR f:SerialDataFifo;
                        VAR d:DataRec);
BEGIN

  WITH f.Ovd DO
  BEGIN                    {teste pile vide}
    IF Count = 0 THEN
    BEGIN                  {si oui, attend}
      waitfor := addr (NotEmpty);
      wait;
    END;
                                {quand donnée disponible}
    Count:=Count-1;             {décrémente count}
    d :=f.data[Outptr];         {lit la donnée}
    Outptr:=Outptr+1;           {incrémente pointeur sortie}
    IF Outptr > SerialDataFifoSize THEN
      Outptr:=1;                {"enroule" ptr si nécessaire}

    {si une tache attend après cette fifo}
    {elle est réveillée}
    IF NotFull <> NIL THEN
      send(NotFull);
  END;

END;


PROCEDURE PutDisplayData (d:DisplayRec;
                      VAR f:DisplayFifoType);

BEGIN

  WITH f.Ovd DO
  BEGIN                               {teste pile pleine}
    IF Count = DisplayFifoSize THEN
    BEGIN                             {si oui, attend}
      waitfor := addr (NotFull);
      wait;
    END;

    Count:=Count+1;                   {incrémente count}
    f.data[Inptr] :=d;                {stocke la donnée}
    Inptr:=Inptr+1;                   {incrémente pointeur entrée}
    IF Inptr > DisplayFifoSize THEN
      Inptr:=1;                      {"enroule" ptr si nécessaire}

    {si une tache attend après cette fifo}
    {elle est réveillée}
    IF NotEmpty <> NIL THEN
      send(NotEmpty);
    END;
END;


PROCEDURE GetDisplayData(VAR f:DisplayFifoType;
                         VAR d:DisplayRec);

BEGIN

  WITH f.Ovd DO
  BEGIN                               {teste pile vide}
    IF Count = 0 THEN
    BEGIN                             {si oui attend}
      waitfor := addr (NotEmpty);
      wait;
    END;
                                      {quand donnée disponible}
    Count:=Count-1;                   {décrémente count}
    d :=f.data[Outptr];               {lit la donnée}
    Outptr:=Outptr+1;                 {incrémente pointeur sortie}
    IF Outptr > DisplayFifoSize THEN
      Outptr:=1;                      {"enroule" ptr si nécessaire}
      {si une tache attend après cette fifo}

      {elle est réveillée}
      IF NotFull <> NIL THEN
        send(NotFull);

    END;

  END;

{Inclus le système de menu}

{$I Menu.pas}

{Inclus les procédures série}

{$I serial.pas}

{********* Procédures Série Additionnelles **********}
  PROCEDURE SetBreak (PortAddr:Integer; State:Boolean);

{Contrôle la génération d'un break}
{pour un port COM donné}

  VAR

    Temp:  Byte;

  BEGIN

{Lit le registre de contrôle de ligne}
{du 8250, et positionne ou efface le}
{bit 6 de break comme spécifié.}
{Ecrit la nouvelle valeur}
{dans le registrede contrôle}

    Temp := port [PortAddr+LineControl] ;
    IF State THEN
      Temp := Temp OR $40
    ELSE
      Temp := Temp AND $BF;
    port[PortAddr+LineControl] := Temp;

  END;


  PROCEDURE MakeHandshake(PortAddr:Integer;
                          Status:Byte);

  VAR

    Temp:  Byte;

  BEGIN

{Modifie les bits du registre de contrôle}
{du modem du port COM spécifié, en fonction}
{des bits de la variable Status.}
{Cette procédure est utilisée pour}
{forcer les lignes de handshake du}
{port COM à être d'accord}

    Temp := port[PortAddr+ModemControl];

    IF (Status AND BrkBit) <> 0 THEN
      SetBreak(PortAddr,True)
    ELSE
      SetBreak(PortAddr,False);
    IF (Status AND CTSBit) <> 0 THEN
      Temp := Temp OR $02
    ELSE
      Temp := Temp AND $FD;

    IF(Status AND DSRBit) <> 0 THEN
      Temp := Temp OR $01
    ELSE
      Temp := Temp AND $FE;

{Stocke la nouvelle valeur du registre ModemControl}

    port [PortAddr+ModemControl] := Temp;

  END;


{************** Procedures d'affichage ***********}

PROCEDURE BuildDisplay;

{Initialise l'écran principal du SPA}
{Les variables OldCOM1 et OldCOM2 sont}
{consciencieusement démolies pour forcer}
{l'écran à être mis-à-jour, après qu'il}
{soit initialisé}

BEGIN

  OldCOM1_Status := $FF;
  OldCOM2_Status := $FF;

  WriteStringAt(Line1, Low,1,1);
  WriteStringAt(Line2, Low,1,2);
  WriteStringAt(Line3, Low,1,3);
  WriteStringAt(Line4, Low,1,4);
  WriteStringAt(Line5, Low,1,5);
  WriteStringAt(Line6, Low,1,6);
```

```
  WriteStringAt(Line24,Rev,1,24);

END;


PROCEDURE DisplayHandShakeStatus  (PortAddr:Integer;
                                   InStatus,
                                   OutStatus:Byte);

CONST

  StatLineNum = 3;  {ligne status}
  InLineNum   = 4;  {ligne du protocole en entrée}
  OutLineNum  = 5;  {ligne du protocole en sortie}

  CDOffset  = 11;   {numéro de colonne des}
  RIOffset  = 16;   {différents éléments à afficher}
  DSROffset = 22;
  CTSOffset = 28;
  BRKOffset = 13;
  FEOffset  = 18;
  PEOffset  = 23;
  OROffset  = 28;
  DTROffset = 12;
  RTSOffset = 22;

VAR

  DisplayOffset:  Integer;
  Ind:            Char;

BEGIN

{Une donnée est affichée sur l'écran}
{Sa position d'affichage est déterminée}
{par le numéro du port COM d'où elle}
{provient. Les données COM1 sont affichées}
{à l'offset 0, les données de COM2}
{sont affichées à l'offset 51}

  IF PortAddr = COM1 THEN
    DisplayOffset := 0
  ELSE
    DisplayOffset := 51;

  IF(InStatus AND BrkBit) <> 0 THEN
    Ind := 'B'
  ELSE
    Ind := '-';
  WriteStringAt(Ind,High,
                BRKOffset+DisplayOffset,
                StatLineNum);

  IF(InStatus AND FEBit) <> 0 THEN
    Ind := 'E'
  ELSE
    Ind := '-';
  WriteStringAt(Ind,High,
                FEOffset+DisplayOffset,
                StatLineNum);

  IF(InStatus AND PEBit) <> 0 THEN
    Ind := 'E'
  ELSE
    Ind := '-';
  WriteStringAt(Ind,High,
                PEOffset+DisplayOffset,
                StatLineNum);

  IF(InStatus AND ORBit) <> 0 THEN
    Ind := 'E'
  ELSE
    Ind := '-';
  WriteStringAt(Ind,High,
                OROffset+DisplayOffset,
                StatLineNum);

  IF(InStatus AND CDBit) <> 0 THEN
    Ind := 'M'
  ELSE
    Ind := 'S';
  WriteStringAt(Ind,High,
                CDOffset+DisplayOffset,
                InLineNum);

  IF(InStatus AND RIBit) <> 0 THEN
    Ind := 'M'
  ELSE
    Ind := 'S';
  WriteStringAt(Ind,High,
                RIOffset+DisplayOffset,
                InLineNum);

  IF(InStatus AND DSRBit) <> 0 THEN
    Ind := 'M'
  ELSE
    Ind := 'S';
  WriteStringAt(Ind,High,
                DSROffset+DisplayOffset,
                InLineNum);

  IF(InStatus AND CTSBit) <> 0 THEN
    Ind := 'M'
  ELSE
    Ind := 'S';
  WriteStringAt(Ind,High,
                CTSOffset+DisplayOffset,
                InLineNum);


{Note : L'autre port COM est consulté en}
{fonction du status de sortie.}
{DTR de ce port doit être égal à DSR}
{de l'autre port. RTS de ce port doit}
{être égal à CTS de l'autre port.}

  IF(OutStatus AND DSRBit) <> 0 THEN
    Ind := 'M'
  ELSE
    Ind := 'S';
  WriteStringAt(Ind,High,
                DTROffset+DisplayOffset,
                OutLineNum);

  IF(OutStatus AND CTSBit) <> 0 THEN
    Ind := 'M'
  ELSE
    Ind := 'S';
  WriteStringAt(Ind,High,
                RTSOffset+DisplayOffset,
                OutLineNum);

END;


PROCEDURE DisplayBufferStatus;

VAR

PerCentage:  Integer;
PerCentStr:  Str80;
BEGIN

  PerCentage := Round((COM1_Input_Fifo.Ovd.Count/
                       SerialDataFifoSize) * 100);
  Str(PerCentage:2,PerCentstr) ;
  WriteStringAt(PerCentstr + '%',High,12,2);

  PerCentage := Round((COM1_Output_Fifo.Ovd.Count/
                       SerialDataFifoSize) * 100);
  Str(PerCentage:2,PerCentstr) ;
  WriteStringAt(PerCentstr + '%',High,26,2);

  PerCentage := Round((COM2_Input_Fifo.Ovd.Count/
                       SerialDataFifoSize) * 100);
  Str(PerCentage:2,PerCentstr) ;
  WriteStringAt(PerCentstr + '%',High,63,2);

  PerCentage := Round((COM2_Output_Fifo.Ovd.Count/
                       SerialDataFifoSize) * 100);
  Str(PerCentage:2,PerCentstr) ;
  WriteStringAt(PerCentstr + '%',High,77,2);

  PerCentage := Round((DisplayFifo.Ovd.Count/
                       DisplayFifoSize) * 100);
  Str(PerCentage:2,PerCentstr) ;
  WriteStringAt(PerCentstr + '%',High,47,5);

END;

FUNCTION FormatCharAscii (Num:Integer) : Str3;

BEGIN

  IF Num < ORD(' ') THEN
    FormatCharAscii := AsciiStrs[Num]
  ELSE
    FormatCharAscii := chr(Num);

END;

FUNCTION FormatCharHex (Num: Integer) : Str3;

BEGIN

  FormatCharHex :=
    HexDigits[(Num SHR 4) AND $000F + 1 ] +
    HexDigits[(Num AND $000F) + 1];

END;

FUNCTION DisplayData (D:DisplayRec) : Integer;

VAR

  VideoAttrib:  AttribType;
  Temp:         STRING[7];

BEGIN

  WITH D DO
  BEGIN
    {si des données d'un des deux canaux}
    {doivent etre affichées, alors...}

    IF(((Tag = FromCOM1) AND COM1_Display_Data) OR
      ((Tag = FromCOM2) AND COM2_Display_Data)) THEN
    BEGIN
      {modifie les attributs vidéo en}
      {fonction du canal série dont les}
      {données proviennent}
```

```
      IF Tag = FromCOM1 THEN

        VideoAttrib := Low
      ELSE
        VideoAttrib := Rev;

      {affichage ASCII ou hexa}
      IF AsciiDisplay THEN
        Temp := FormatCharAscii(DR.Data)
      ELSE
        Temp := FormatCharHex(DR.Data);

      IF HandShakeDisplay THEN
        Temp := Temp + ':$'+
          FormatCharHex(DR.Status);

      {écrit la chaine des données série}
      WriteString (Temp, VideoAttrib) ;

      {si formatage avec des espaces}
      IF AddSpace THEN
      BEGIN
      {affiche un espace, et ajoute}
      {un espace dans la chaine}
      {pour le calcul de sa longueur}
        write(' ');
        Temp := Temp + ' ';
      END;

      {retourne longueur des données formatées}
      DisplayData := length(Temp);
    END
    ELSE
      {si pas de données, retourne longueur 0}
      DisplayData := 0;
  END;

END;

{********** Procedures diverses **********}

PROCEDURE Init_Program;

{Initialisation de l'analyseur de protocole}

BEGIN

  Init_Fifos;

  COM2_Data_Acquire := True;
  COM2_Data_Acquire := True;

  COM1_Display_Data := True;
  COM2_Display_Data := True;
  TriggerEnabled    := False;
  COM1_Is_Triggered := False;
  COM2_Is_Triggered := False;

  AsciiDisplay      := True;
  HandShakeDisplay  := False;
  AddSpace          := True;

  COM1_Status       := 0;
  COM2_Status       := 0;

  {valeurs par défaut : 1200 bauds, 8 bits}
  {de données, 1 stopbit, pas de parité}
  {les deux ports COM ont toujours les}
  {mêmes paramètres}

  COM_Rate      := 8; {1200 baud}
  COM_StopBits  := 1;
  COM_DataBits  := 8;
  COM_Parity    := none;

  SetNewCOMParameter;       {applique les paramètres}

END;


PROCEDURE VerifyHardware;

CONST

  TestByte = $5A;           {essaie d'écrire puis de}
                            {relire cette valeur}
  SafeByte = $03;           {par défaut : 8 bits, 1 stop}

BEGIN

  ClrScr;
  {seulement en cas de réentrance}
  Disable_Serial_Devices;

  WriteString('  Serial Protocol Analyzer. Vérification du Hardware ' ,Rev);
  WriteStringAt('Test de COM1',HighBlink,2,3);
  port[COM1+LineControl] := TestByte;
  delay(700);
  IF port[COM1+LineControl] <> TestByte THEN
  BEGIN
    WriteStringAt('Port COM1 défectueux ou manquant',
                  Rev,2,3);
    Halt;
  END;
  port[COM1+LineControl] := SafeByte;
  WriteStringAt('Port COM1 Vérifié avec succès',
                Low,2,3);

  delay(700);
  WriteStringAt('Test de COM2',HighBlink,
                2,4);
  port[COM2+LineControl] := TestByte;
  delay(500);
  IF port[COM2+LineControl] <> TestByte THEN
  BEGIN
    WriteStringAt('Port COM2 défectueux ou manquant',
                  Rev,2,4);
    Halt;
  END;
  port[COM2+LineControl] := SafeByte;
  WriteStringAt('Port COM2 Vérifié avec succès',Low,2,4);

  delay (2000);
  ClrScr;

END;


{****** Exécution des commandes du menu *******}

{Pré-déclarée plus haut}

PROCEDURE ProcessCmd;
VAR

  HexStr:   Str3;
  HexChar,
  ErrCode:  Integer;
  Ch:       Char;

BEGIN

  IF cmd_code <> 0 THEN
  BEGIN
    CASE cmd_code OF
      1: ExitMenu := True;
      2: BEGIN                  {300 bauds}
           COM_Rate := 6;
           SetNewCOMParameter;
         END;

      3: BEGIN                  {600 bauds}
           COM_Rate := 7;
           SetNewCOMParameter;
         END;

      4: BEGIN                  {1200 bauds}
           COM_Rate := 8;
           SetNewCOMParameter;
         END;

      5: BEGIN                  {2400 bauds}
           COM_Rate := 11;
           SetNewCOMParameter;
         END;

      6: BEGIN                  {4800 bauds}
           COM_Rate := 13;
           SetNewCOMParameter;
         END;

      7:  BEGIN                  {9600 bauds}
           COM_Rate := 15;
           SetNewCOMParameter;
         END;

      8:  BEGIN                  {1 stop bit}
           COM_StopBits := 1;
           SetNewCOMParameter;
         END;

      9:  BEGIN                  {2 stop bits}
           COM_StopBits := 2;
           SetNewCOMParameter;
         END;

      10: BEGIN                  {5 bits données}
           COM_DataBits := 5;
           SetNewCOMParameter;
         END;

      11: BEGIN                  {6 bits données}
           COM_DataBits := 6;
           SetNewCOMParameter;
         END;

      12: BEGIN                  {7 bits sonnées}
           COM_DataBits := 7;
           SetNewCOMParameter;
         END;

      13: BEGIN                  {8 bits données}
           COM_DataBits := 8;
           SetNewCOMParameter;
         END;

      14: BEGIN                  {Parité impaire}
           COM_Parity := Odd;
           SetNewCOMParameter;
         END;

      15: BEGIN                  {Parité paire}
```

```
            COM_Parity := Even;

            SetNewCOMParameter;
          END;

      16: BEGIN                  {Pas de parité}
            COM_Parity := None;
            SetNewCOMParameter;
          END;

      17: BEGIN                  {Affiche COM1 seulement}
            COM1_Display_Data := True;
            COM2_Display_Data := False;
          END;

      18: BEGIN                  {Affiche COM2 seulement}
            COM1_Display_Data := False;
            COM2_Display_Data := True;
          END;

      19: BEGIN                  {Affiche COM1 & COM2}
            COM1_Display_Data := True;
            COM2_Display_Data := True;
          END;

      20: BEGIN                  {point d'arrêt sur COM1}
            GoToXY(1,25);
            ClrEol;
            COM1_Is_Triggered := True;
            COM2_Is_Triggered := False;
            WriteStringAt('COM1 awaiting trigger',
                          HighBlink,3,25);
          END;

      21: BEGIN                  {point d'arrêt sur COM2}
            GoToXY(1,25);
            ClrEol;
            COM1_Is_Triggered := False;
            COM2_Is_Triggered := True;
            WriteStringAt('COM2 awaiting trigger',
                          HighBlink,3,25);
          END;

      22: BEGIN                   {Saisit la donnée du point d'arrêt}
            GoToXY(1,MenuLine4);  {position du curseur}
            ClrEol;               {efface ligne}
            WriteString('Entrée la donnée à scruter en hexa sur deux
chiffres: '
                        ,High);

            {initialise str avec le préfixe hexa}
            HexStr := '$';
            FOR HexChar := 1 TO 2 DO
            BEGIN
              REPEAT
                {lit un caractère}
                Ch := upcase(Char(GetKey));
                {jusqu'à ce qu'un caractère valide soit lu}
              UNTIL pos(Ch,HexDigits) <> 0;
              {affiche le caractère}
              write (Ch);
              {ajoute le caractère hexa à la chaine}
              HexStr := HexStr + Ch;
            END;
            {converti hexa en entier}
            Val(HexStr,TriggerPattern,ErrCode) ;
            GoToXY(28,25);
            Write ('Valeur à scruter: ',
                  FormatCharHex (TriggerPattern) ) ;
          END;

      23: BEGIN          {Affichage JUSQU'AU déclenchement}
            TriggerMode := Before;
            WriteStringAt('Mode: Affichage Avant déclenchement'
                          ,Low,51,25);
            IF COM1_Is_Triggered THEN
              {on commence en affichant les données}
              COM1_Display_Data := True
            ELSE
            IF COM2_Is_Triggered THEN
              COM2_Display_Data := True
            ELSE
              WriteStringAt('Erreur de mode - port COM'
                            ,HighBlink,51,25);
          END;

      24: BEGIN                   {Affiche APRES déclenchement}
            TriggerMode := After;
            WriteStringAt('Mode: Affichage Après déclenchement'
                          ,Low,51,25);
            IF COM1_Is_Triggered THEN
              {on commence sans afficher les données}
              COM1_Display_Data := False
            ELSE
            IF COM2_Is_Triggered THEN
              COM2_Display_Data := False
            ELSE
              WriteStringAt('Erreur de mode - port COM'
                            ,HighBlink,51,25);
          END;

      25: TriggerEnabled := True;        {autorise déclenchement}

      26: BEGIN                          {stoppe déclenchement}
            TriggerEnabled := False;
            {stoppe déclenchement pour les deux ports}
            COM1_Is_Triggered := False;

            COM2_Is_Triggered := False;

            {autorise l'affichage des données sur les deux ports}
            COM1_Display_Data := True;
            COM2_Display_Data := True;
            GoToXY(3,25);
            ClrEol;
            WriteString('Déclenchement interdit',Low);
          END;

      27: BEGIN                  {Affichage des données Ascii}
            AsciiDisplay := True;
            HandShakeDisplay := False;
          END;

      28: BEGIN                  {Affichage des données Ascii avec protocole}
            AsciiDisplay := True;
            HandShakeDisplay := True;
          END;

      29: BEGIN                  {Affichage des données Hexa}
            AsciiDisplay := False;
            HandShakeDisplay := False;
          END;

      30: BEGIN                  {Affichage des données Hexa avec protocole}
            AsciiDisplay := False;
            HandShakeDisplay := True;
          END;

      31: BEGIN                   {Mode affichage avec/sans espaces}
            AddSpace := NOT AddSpace;
          END;

      32: BEGIN                   {Début d'acquisition des données}
            COM1_Data_Acquire := True;
            COM2_Data_Acquire := True;
          END;

      33: BEGIN                   {Fin d'acquisition des données}
            COM1_Data_Acquire := False;
            COM2_Data_Acquire := False;
          END;

      34: BEGIN                   {Efface écran}
            {claim the screen}
            Alloc(ScreenAccess);
            ClrScr;
            DrawMenuFrame;
            {set cursor postion to home}
            OldXPos := 1;
            OldYPos := 1;
            Dealloc(ScreenAccess);
          END;

      35: Init_Program;          {réinitialise l'analyseur}

      36: BEGIN                   {Quitte le programme}
            ExitMenu := True;
            ExitProgram := True;
          END;


    END;
  END;
END;
{************ Tâches *************}

PROCEDURE ProcessKeysTask;

VAR

ch:   Char;

Done,
OldCOM1Flag,
OldCOM2Flag:  Boolean;

BEGIN

  Done := False;

  REPEAT

    {lit une touche au clavier}

    Ch := char(GetKey);

    CASE Ch OF

    Esc:  BEGIN                      {Touche Esc}
            {affiche et gère le menu}
            Done := DoMenu;
            {réaffiche écran quand fini}
            BuildDisplay;
          END;

    Sp:   BEGIN                      {Barre espace}
            {Sauve ancien état d'affichage}
            OldCOM1Flag := COM1_Display_Data;
            OldCOM1Flag := COM2_Display_Data;

            {arrête de remplir la pile d'affichage}
            {pour ne pas qu'elle déborde pendant}
```

```
          {la pause. stoppe l'affichage des deux}

          {ports série}
          COM1_Display_Data := False;
          COM2_Display_Data := False;
          GoToXY(1,24);
          ClrEol;
          GoToXY(23,24);
          Writestring(
            'Appuyez sur <Entrée> pour continuer'
            ,HighBlink);
          REPEAT
            {boucle jusqu'à ce qu'un appui sur Entrée}
            {reprenne l'affichage}
            Ch := Char(GetKey);
          UNTIL Ch = Cr;

          {restaure l'état d'affichage}
          COM1_Display_Data := OldCOM1Flag;
          COM2_Display_Data := OldCOM2Flag;
          BuildDisplay;                    {réaffiche l'écran}
        END;
      END;

  UNTIL Done;

END;


PROCEDURE MoveCOM1Data;

{Lit les données sur le port COM1 et les}
{écrit sur le port COM2 et dans la pile}
{d'affichage si l'affichage est autorisé}
VAR

  Sd: DataRec;
  Dd: DisplayRec;

BEGIN

  REPEAT
    {exécute les autres tâches}
    {s'il n'y a pas de données}
    WHILE COM1_Input_Fifo.Ovd.Count = 0 DO
      yield;
    {quand des données sont lue}
    {il faut les déplacer}

    {supprime les interruption, lit les}
    {données, et restaure les interruptions}
    INLINE($FA);
    GetSerialData(COM1_Input_Fifo,Sd);
    INLINE($FB);

    {écrit les données dans la pile FIFO de COM2}
    PutSerialData(Sd,COM2_Output_Fifo);
    {si un déclencheur est actif}
    IF COM1_Is_Triggered AND
      TriggerEnabled THEN
    BEGIN
      {et qu'une correspondance est trouvée}
      IF Sd.Data = TriggerPattern THEN
      {indique que l'évènement est rencontré}
      {supprime le déclencheur car un seul}
      {évènement est attendu}
      BEGIN
        TriggerEnabled := False;
        WriteStringAt('   COM1 Déclenché     ', Low,3,25);
        {Si affichage avant déclencheur}
        {alors stoppe l'affichage}
        IF TriggerMode = Before THEN
          COM1_Display_Data := False
        ELSE
          {Si affichage après le déclenchement}
          {démarre l'affichage}
          COM1_Display_Data := True;

      END;
    END;
    {si on affiche les données...}
    IF COM1_Display_Data THEN
    {recopie les données dans la pile d'affichage...}
    BEGIN
      {marquées avec le numéro du port}
      Dd.DR := Sd;
      Dd.Tag := FromCOM1;
      PutDisplayData (Dd, DisplayFifo) ;
    END;

  UNTIL False;

END;


PROCEDURE MoveCOM2Data;

{Lit les données sur le port COM2 et les}
{écrit sur le port COM1 et dans la pile}
{d'affichage si l'affichage est autorisé}

VAR

  Sd: DataRec;
  Dd: DisplayRec;

BEGIN

  REPEAT
    {exécute les autres tâches}
    {s'il n'y a pas de données}

    WHILE COM2_Input_Fifo.Ovd.Count = 0 DO
      yield;

    {quand des données sont lue}
    {il faut les déplacer}

    {supprime les interruption, lit les}
    {données, et restaure les interruptions}
    INLINE($FA);
    GetSerialData(COM2_Input_Fifo,Sd);
    INLINE($FB);

    {écrit les données dans la pile FIFO de COM1}
    PutSerialData(Sd,COM1_Output_Fifo);

  {si un déclencheur est actif}
  IF COM2_Is_Triggered AND
    TriggerEnabled THEN
  BEGIN
    {et qu'une correspondance est trouvée}
    {indique que l'évènement est rencontré}
    {supprime le déclencheur car un seul}
    {évènement est attendu}
    IF Sd.Data = TriggerPattern THEN
    BEGIN
      TriggerEnabled := False;
      WriteStringAt('   COM2 Déclenché     ', Low,3,25);
      {Si affichage avant déclenchement}
      {alors stoppe l'affichage}
      IF TriggerMode = Before THEN
        COM2_Display_Data := False
      ELSE
        {Si affichage après le déclenchement}
        {démarre l'affichage}
        COM2_Display_Data := True;

    END;
  END;
  {si on affiche les données...}
  IF COM2_Display_Data THEN
  BEGIN
    {recopie les données dans la pile d'affichage...}
    {marquées avec le numéro du port}
    Dd.DR := Sd;
    Dd.Tag := FromCOM2;
    PutDisplayData(Dd,DisplayFifo);
  END;

  UNTIL False;

END;


PROCEDURE OutputCOM1Data;

{Lit les données dans la pile FIFO}
{en sortie du port COM1 et les}
{écrit sur le port COM1}
CONST

{au plus 80 caractères sont écrits sur}
{le port série avant une commutation de taches}

OutputsPerYield = 80;


VAR

Outs:   Integer;
Sd:     DataRec;
Urgent: Boolean;

BEGIN

  REPEAT

    {exécute jusqu'à ce que des données}
    {soient disponibles dans la pile}
    WHILE (COM1_Output_Fifo.Ovd.Count = 0) DO
    BEGIN
      {accorde le protocole de COM1 et COM2}
      {puis commute les taches}
      MakeHandshake(COM1,COM2_Status);
      yield;
    END;

    {Il y a des données dans la pile}
    {c'est urgent si la pile est à moitié pleine}

    Urgent := (COM1_Output_Fifo.Ovd.Count >= (SerialDataFifoSize / 2) );

    {Initialise compteur à la valeur max}
    Outs := OutputsPerYield;

    {Tant qu'il y a des données à transmettre}

    WHILE ((COM1_Output_Fifo.Ovd.Count <> 0) AND (Outs <> 0)) DO
    BEGIN

      {teste si l'UART est prêt à transmettre}
```

```
    WHILE(Get_Serial_Status(COM1) AND TxRdyBit) = 0 DO
    BEGIN
      {si on a du temps, commute en attendant}
      If NOT Urgent THEN
        yield;
    END;
    {Lit les données à écrire}
    {Prépare le protocole et les données}

    GetSerialData(COM1_Output_Fifo,Sd);
    MakeHandshake(COM1,Sd.Status);
    port [COM1] := Sd.Data;

    {Un de plus en moins}
    Outs := Outs - 1;

  END;
  yield;

UNTIL False;

END;



PROCEDURE OutPutCOM2Data;

{Lit les données dans la pile FIFO}
{en sortie du port COM1 et les}
{écrit sur le port COM1}
CONST

{au plus 80 caractères sont écrits sur}
{le port série avant une commutation de taches}

  OutPutsPerYield = 80;

VAR

  Outs:   Integer;
  Sd:     DataRec;
  Urgent: Boolean;

BEGIN

  REPEAT

    {exécute jusqu'à ce que des données}
    {soient disponibles dans la pile}
    WHILE (COM2_Output_Fifo.Ovd.Count = 0) DO
    BEGIN
      {accorde le protocole de COM1 et COM2}
      {puis commute les taches}
      MakeHandshake(COM2,COM1_Status);
      yield;
    END;

    {Il y a des données dans la pile}
    {c'est urgent si la pile est à moitié pleine}

    Urgent := (COM2_Output_Fifo.Ovd.Count >= (SerialDataFifoSize / 2) );

    {Initialise compteur à la valeur max}

    Outs := OutputsPerYield;

    {Tant qu'il y a des données à transmettre}

    WHILE ((COM2_Output_Fifo.Ovd.Count <> 0) AND (Outs <> 0)) DO
    BEGIN
      {Teste si l'UART est prêt à transmettre}

      WHILE (Get_Serial_Status(COM2) AND TxRdyBit) = 0 DO
      BEGIN
        {si on a du temps, commute en attendant}
        IF NOT Urgent THEN
        yield;
      END;

      {Lit les données à écrire}
      {Prépare le protocole et les données}

      GetSerialData(COM2_Output_Fifo,Sd);
      MakeHandshake(COM2,Sd.Status);
      port[COM2] := Sd.Data;

      {un de moins à écrire}
      Outs := Outs - 1;

    END;
    yield;

  UNTIL False;
END;


PROCEDURE DisplayCOMData;

CONST

{nombre max de données à afficher avant de commuter}
MaxDisplayItems = 20;

VAR
  DisplayItems,

  DataLen:      Integer;
  D:            DisplayRec;
BEGIN

  {curseur positionné en haut}
  {à gauche de la fenêtre d'affichage}
  OldXPos := 1;
  OldYPos := 1;

  REPEAT
    {s'il y a des données dans la pile d'affichage}
    IF DisplayFifo.Ovd.Count <> 0 THEN
    BEGIN
      {demande l'accès à l'écran, définit la fenêtre}
      {positionne le curseur et initialise le}
      {compteur d'éléments à afficher}
      Alloc(ScreenAccess);
      Window(1,7,80,23);
      GoToXY(OldXPos,OldYPos);
      DisplayItems := MaxDisplayItems;

      WHILE ((DisplayFifo.Ovd.Count <> 0 ) AND (DisplayItems <> 0)) DO
      {tant qu'il y a des donénes à afficher}
      BEGIN
        {lit la donnée, l'affiche, décrémante}
        {le compteur de données. Sauve la}
        {longueur de l'élément affiché}

        GetDisplayData(DisplayFifo,D);
        DataLen := DisplayData(D);
        DisplayItems := DisplayItems - 1;
        {s'il n'y a plus de place pour un autre}
        {élément de la même longueur et que l'on}
        {est sur la dernière ligne de la fenêtre,}
        {alors positionne le curseur au début de}
        {la fenêtre et écrase les anciennes données}
        {pour éviter un scroll de la fenêtre.}
        {Si l'on n'est pas sur la dernière}
        {ligne de la fenêtre, passe au début}
        {de la ligne suivante. Dans les deux}
        {cas, la ligne doit être effacée.}
        IF WhereX + DataLen >= 80 THEN
          IF WhereY = 17 THEN
            GoToXY(1,1)
          ELSE
            WriteLn;
        ClrEol;
      END;
      {sauve nouvelle position du curseur}
      OldXPos := WhereX;
      OldYPos := WhereY;
      {quitte le mode fenêtre, libère}
      {l'écran et commute la tache}
      Window(1, 1, 80, 25);
      Dealloc(ScreenAccess);
    END
    ELSE
      {s'il n'y a pas de données à écrire}
      {commute simplement la tache}
      yield;
  UNTIL False;
END;


PROCEDURE Timer;

VAR

  Toggle:  Boolean;

  BEGIN
    Toggle := True;

    REPEAT

    {toutes les demi secondes}
    pause(4);

    {si l'écran doit être mis-à-jour}
    IF UpdateScreenStatus THEN
    BEGIN
      {verrouille l'écran}
      Alloc(ScreenAccess);

      {si le status d'un des ports a changé}
      IF ((COM1_Status <> OldCOM1_Status) OR (COM2_Status <> OldCOM2_Status) )
THEN
      {met à jour l'écran}
      BEGIN
        {affiche le nouvel état de COM1}
        DisplayHandshakeStatus(COM1,COM1_Status, COM2_Status);
        {sauve le nouvel état}
        OldCOM1_Status := COM1_Status;

        {affiche le nouvel état de COM2}
        DisplayHandShakeStatus(COM2, COM2_Status, COM1_Status);
        OldCOM2_Status := COM2_Status;

      END;

      {toutes les secondes, met à jour}
      {l'état du buffer écran}
      IF Toggle THEN
        DisplayBufferStatus;
```

```
      {déverrouille l'écran}
      Dealloc(ScreenAccess);
      Toggle := NOT Toggle;
    END;

  UNTIL False;

END;


BEGIN {main programme principal}

  {vérifie la présence de COM1 et COM2}
  VerifyHardware;

  {initialise le multitache}
  Init_Kernel;

  {initialise la structure des menus}
  Init_Menu;

  {installe les routines d'interruption série ISR}
  Install_Serial_Handlers;

  {Initialise les chaines utilisées pour l'affichage}
  Line1 := Char(201)+' COM1 '+repl(16,Char(205))+
           ' Analyseur de Protocol Série Ver: 1.0 '+
           repl(15,Char(205))+ ' COM2 '+Char(187);

  Line2 := Char(186)+ ' In Fifo:     Out Fifo:     ' +
           Char(186)+ repl(10,' ')+'By '+repl(10,' ') +
           Char(186)+ ' In Fifo:     Out Fifo:     ' +
           Char(186);

  Line3 := Char(186) + ' Stat- BRK:  FE:  PE:  OR: ' +
           Char(186) + '   Craig A. Lindley   ' +
           Char(186) + ' Stat- BRK:  FE:  FE:  OR: ' +
           Char (186);

  Line4 := Char(186) + ' In - CD:  RI:  DSR:  CTS: ' +
           Char(186) + repl(22,' ') +
           Char(186) + ' In - CD:  RI:  DSR:  CTS: ' +
           Char(186);

  Line5 := Char(186) + ' Out- DTR:      RTS:' + repl(7,' ') +
           Char(186) + '   Fifo Affichage:    ' +
           Char(186) + ' Out- DTR:      RTS:' + repl(7,' ') +
           Char(186);

  Line6 := Char(200) + repl(27,Char(205)) +
           Char(202) + repl(22,Char(205)) +
           Char(202) + repl(27,Char(205)) +
           Char(188);

  Line24 := ' <Esc> Menu      <Space Bar> Pause       COM1 - Normal : COM2 -
Vidéo inverse';

  BuildDisplay;   {Affiche écran principal}
  Init_Program;   {initialise les variables}

  {initialise le sémaphore de l'écran}
  Initialize_Semaphore(ScreenAccess);

  {autorise la mise à jour de l'état}
  UpdateScreenStatus := True;

  {Duplique les taches}
  Fork;

  IF child_process THEN
    MoveCOM1Data;

  Fork;

  IF child_process THEN
    MoveCOM2Data;

  Fork;

  IF child_process THEN
    OutputCOM1Data;

  Fork;

  IF child_process THEN
    OutputCOM2Data;

  Fork;

  IF child_process THEN
    DisplayCOMData;

  Fork;

  IF child_process THEN
    Timer;


  ProcessKeysTask;

  ClrScr;
  Disable_Serial_Devices;
  Remove_Serial_Handlers;
END.
```

LES CURIEUX SONT SERVIS !

DISK EXPLORER

l'éditeur de tout support magnétique

- Editez un secteur, une piste, un cluster ou tout simplement un fichier sous forme ASCII, héxadécimale ou binaire.
- Changez les attributs des fichiers pour accéder aux données "cachées", modifiez les enregistrements verrouillés en "lecture seulement", récupérez les fichiers perdus
- Modifiez le répertoire, la FAT, l'emplacement des fichiers, les zones de synchronisation, les données, les formats des secteurs et des pistes, et même les registres du contrôleur.
- Intervenez sur des disques de tous formats (5"1/4, 3"1/2, disques durs ou virtuels) et de tous systèmes d'exploitation (MS-DOS, PC-DOS, OS/2, P-System, QNX, CP/M, PICK, PROLOGUE).
- Analysez les encodages ou les "protections", créez vos propres formats.

QUAID ANALYSER

le "must" des debuggers soft pour PC

- Visualisez toutes les interruptions générées par le programme, ainsi que leurs explications en clair.
- Guettez une zone mémoire et stoppez le programme dès que son contenu change.
- Analysez les étapes de chargement et de démarrage (Boot).
- Editez le contenu de la RAM pour vous y promener comme dans un traitement de texte.

Directement logé en mémoire protégée, **ANALYSER** suit pas à pas l'exécution d'un programme et ce, même en l'absence des sources et en dépit des "protections" contre l'analyse et le suivi.

Manuels en français, logiciels non "protégés".

BON DE COMMANDE

..... ex. du DISK EXPLORER au prix de 1012 F HT l'unité (1200 F TTC + 45 F de port)

..... ex du QUAID ANALYSER au prix de 1644 F HT l'unité (1950 F TTC + 45 F de port)

..... ex du QUAID PACK (Explorer&Analyser) 1897 F HT l'unité (2250 F TTC + 35 F de port)

❑ 5"1/4 ❑ 3"1/2

❑ Cheque joint

❑ VISA ❑ EUROCARD ❑ MASTERCARD

N° Exp

❑ Contre remboursement (+ 40 F de frais)

Nom

Adresse

Tél

Code postal Ville

Signature Date

COMPUMART

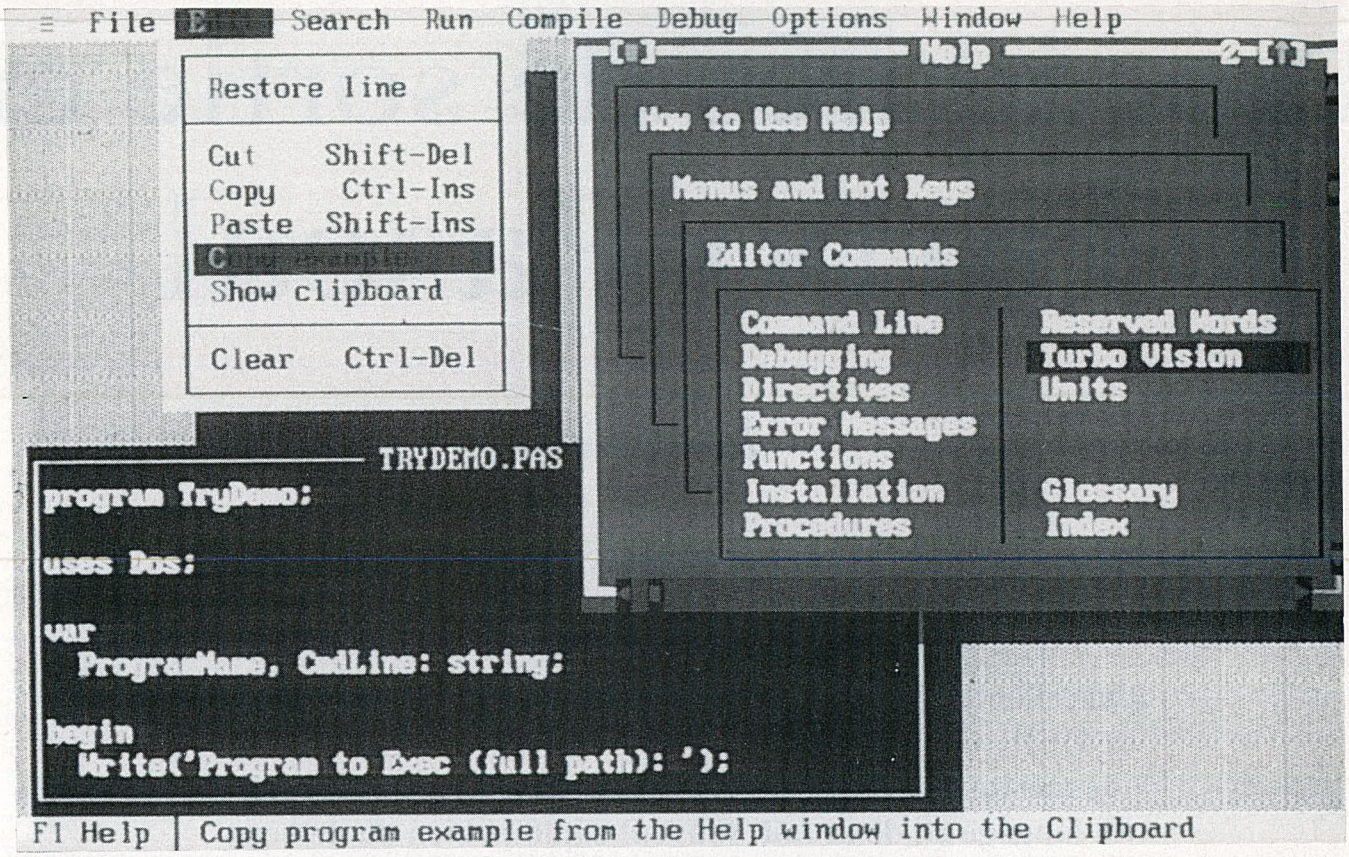
21, rue de Fécamp
75012 Paris

Procédures de sortie en Turbo Pascal

la suite

par Serge Grun

L'excellent article de L. Chambaret paru dans le numéro 6 de DEBUG ("Procédures de sortie en Turbo Pascal") a eu le mérite de présenter clairement un aspect peu connu du langage et, de ce fait, souvent exploité bien au dessous de ses possibilités. Ayant eu à faire usage de ces procédures dans mes propres programmes, je me suis souvent heurté au principal problème qu'elles posent, à savoir qu'il n'est pas possible de retourner au programme appelant après une erreur. Du moins, c'est ce qui est écrit dans le Manuel de Référence de Turbo Pascal.



En fait, il est tout à fait possible d'écrire une procédure de sortie "réentrante", qui permet en quelque sorte au programme de retomber sur ses pieds après un "plantage". Le moyen le plus simple de retourner en arrière est de replacer les registres du processeur dans l'état dans lequel ils étaient dans une circonstance connue. C'est en quelque sorte l'équivalent, en Pascal, des fonctions setjump() et longjump() du C. Si les registres vitaux du processeur sont sauvegardés à un moment donné du déroulement d'un programme, il suffira de les recharger à partir de n'importe quelle routine appelée par la suite pour revenir au point de sauvegarde. Ce mécanisme a toutefois une restriction: il n'est possible de sauter que vers un niveau supérieur d'appel de fonction. Cette exigence est facilement compréhensible. A chaque fois qu'un appel de procédure est fait, c'est à dire à chaque fois que l'on descend d'un niveau dans la hiérarchie des appels, le programme passe un certain nombre de paramètres sur la pile (même si la procédure ou fonction appelée n'en prend pas). Or, parmi ces paramètres, se trouve l'adresse de retour de la procédure qui, sans cette adresse, est complètement perdue. Si vous sautez vers un niveau d'appel inférieur, vous allez restituer également le pointeur de pile tel qu'il existait au moment de la sauvegarde des registres. Mais, comme vous n'avez pas parcouru toute la chaîne d'appels de procédures qui mène à ce point dans le déroulement normal du programme, toute une partie de la pile n'aura pas été construite. La fin de la procédure se manifeste alors d'une façon assez brutale...

Inversement, lorsque l'on saute vers un niveau plus élevé, la chaîne d'appels de procédures est court-circuitée et le contenu de la pile simplement ignoré. Dans la majorité des cas, cela se passe bien. La seule exception est le saut entre deux chaînes d'appels distinctes:

```
Program Demo;
  Procedure A;
    Begin
      { Point 1 }
    End;
  Procedure B;
    Begin
      { Point 2 }
    End;
  Begin { Demo }
    A;
    B
  End.
```

Si la sauvegarde des registres est établie au point 1, le saut vers ce point ne pourra se faire qu'à partir des procédures appelées à partir de la procédure A. Il n'est pas possible, par exemple, de sauter vers 1 à partir de 2. Les deux points sont, en effet, au même niveau et, lorsque l'on est au point 2, l'adresse de retour de la procédure A n'existe plus sur la pile.

Le code qui permet de réaliser cette petite gymnastique est relativement simple. Il est écrit en assembleur, mais rassurez-vous, Turbo Pascal seul est suffisant: tout est encodé sous forme de procédures Inline. Les registres sont sauvegardés dans une structure de type Jmp:

```
Type Jmp=Record
       SP,BP,SS:Word
       CS_IP:Pointer
     End;
```

Remarquez que seuls les registres BP, SP et SS sont sauvegardés. Turbo Pascal n'exige pas la sauvegarde des autres registres au sein du code écrit par l'utilisateur. L'exception est constituée par les boucles, où CX sert de compteur, et les (rares) adressages

relatifs où SI et DI sont utilisés (opérations sur les matrices). Evitez donc de sauvegarder les registres dans ces endroits. Par ailleurs, si vous écrivez des programmes TSR (résidents), vous pouvez être amené à changer le segment de la pile. Pour cette raison, SS est sauvegardé (et que, pour bien faire, les segments devraient l'être aussi). La sauvegarde de DS n'est pas nécessaire: Turbo Pascal ne permet qu'un seul segment de données et, si vous pouvez accéder à la structure Jmp, vous avez déjà, à priori, le bon segment dans le bon registre. La sauvegarde des registres est simple. Il faut seulement recourir à un petit truc pour récupérer CS et IP (qui sont sur la pile):

Le chargement des registres n'est guère plus compliqué. Seule, la récupération de SS s'accompagne de quelques précautions, à savoir, l'inhibition des interruptions :

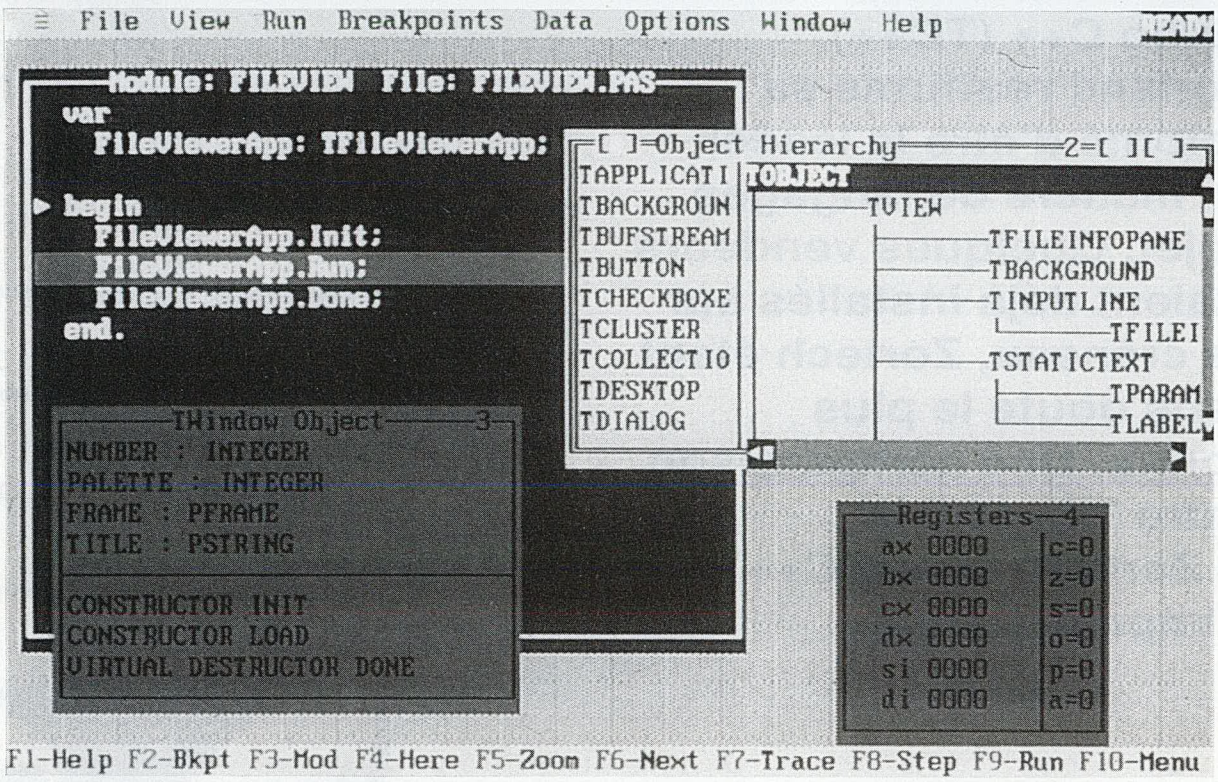
Je vous laisse pour exercice le soin de modifier le code pour sauvegarder les drapeaux (flags).

```
procedure LongJump(var JumpDest:JumpRecord);

  inline(
    $5F/          {pop di          ;di = Ofs(JumpDest)}
    $07/          {pop es          ;es = Seg(JumpDest)}
      $26/$8B/$25/    {mov sp,es:[di]    ;restore sp}
      $26/$8B/$6D/$02/  {mov bp,es:[di+2]  ;restore bp}
      $26/$8B/$45/$04/  {mov ax,es:[di+4]}
    $FA/          {cli}
    $8E/$D0         {mov ss,ax        ;restore ss}
    $FB/          {sti}
      $26/$FF/$6D/$06)/ {jmp far es:[di+6] ;jump far à Jmp.CS_IP}
```

```
procedure SetJump(var JumpDest:Jmp);
  Inline(
    $5F/          {pop di          ;di = Ofs(JumpDest)}
    $07/          {pop es          ;es = Seg(JumpDest)}
      $26/$89/$25/    {mov es:[di],sp    ;save sp}
        $26/$89/$6D/$02/{mov es:[di+2],bp ;save bp}
    $16/          {push ss}
    $58/          {pop ax}
      $26/$89/$45/$04/{mov es:[di+4],ax ;save ds}
      $E8/$00/$00/    {call null       ;push IP onto stack}
                {null:}
    $58/          {pop ax          ;pop into ax}
     $05/$0C/$00/    {add ax,12       ;point to "next:"}
      $26/$89/$45/$06/  {mov es:[di+6],ax ;save jump offset}
      $26/$8C/$4D/$08)/ {mov es:[di+8],cs ;save jump segment}
                {next:}
```

NOTE: Tout le code de cet article est placé dans le domaine public. Les lecteurs de Debug sont explicitement autorisés à en faire l'usage qu'il leur plaira, sous réserve de ne pas m'associer aux conséquences qui pourraient en résulter.

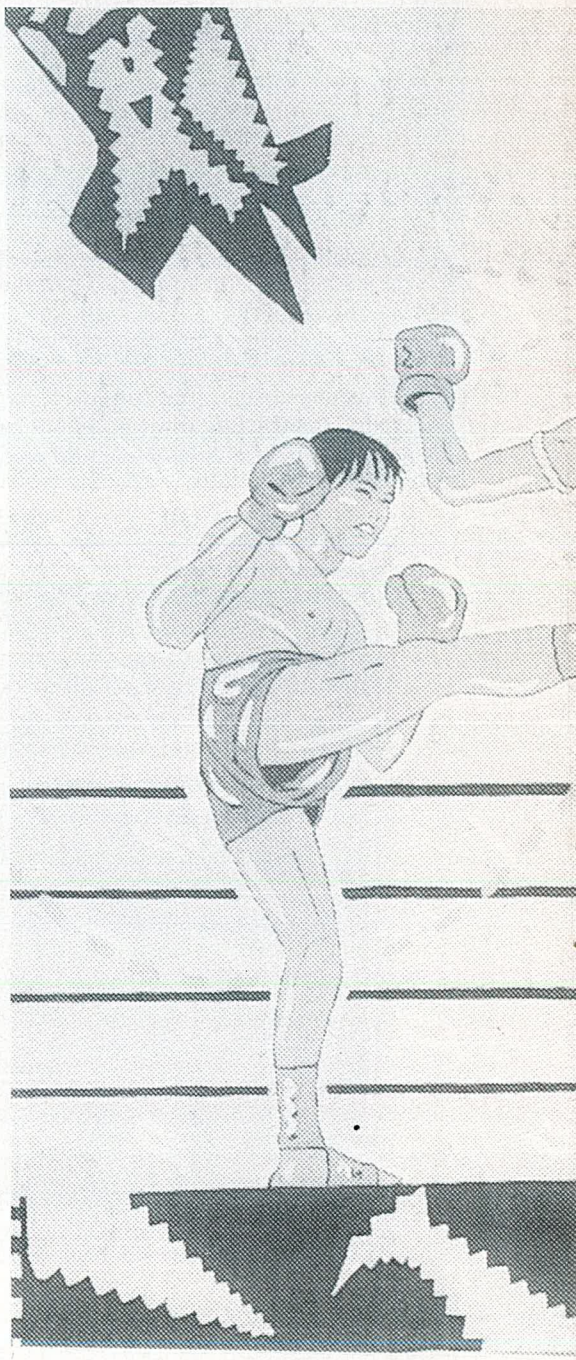


Débogeur Tools

Turbo C++ et

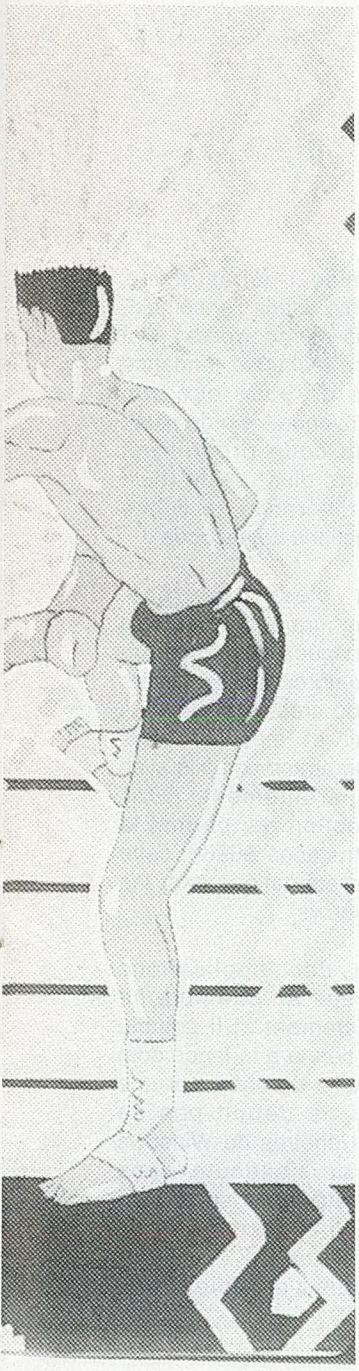
par Serge Grun

La lutte dans le camp des compilateurs C++ natifs promet d'être rude avec l'entrée dans l'arène d'un poids lourd: à travers Turbo C++ (TC++), Borland réaffirme sa volonté d'être le leader en matière de langages objet. Quelques vendeurs étaient déjà installés sur ce créneau, Zortech étant sans doute le plus connu. Une comparaison entre TC++1.0 et le Zortech C++ (ZC++) s'imposait donc.



[SERVICE LECTEURS : INTERESSANT = 28 / MOYEN = 29 / MAUVAIS = 30]

Zortech C++



A ma gauche, le tenant du titre...

La produit examiné en était à la version 2.1 et c'est le kit "Developer's Edition" qui nous a été communiqué. Cette version comprend un environnement de développement intégré, le compilateur C++, plusieurs versions du débogueur, un ensemble de classes C++ et les sources de la librairie. L'ensemble occupe, sous forme compactée, 10 disquettes de 360Ko.

Ajoutez-y 5 manuels (C++ Compiler Ref, C++ Library Ref, C++ Debugger, C++ Tools et C++ 2.1 Update Guide) et quelques notes de dernière minute (le tout en anglais) et vous aurez un outil de programmation C++ complet. L'installation demande environ 3.5 Mo. Le programme inter-actif d'installation crée tous les répertoires nécessaires et n'installe que ce qui lui a été demandé.

ZC++ était jusqu'à présent livré avec l'éditeur ZED. Apartir de la version 2.1, il est accompagné d'un environnement de développement intégré (multi-fenêtres, multi-fichiers, menus déroulants, souris etc.), comparable aux produits Turbo ou Quick, quoique inférieur à mon sens en termes d'ergonomie. Cet environnement existe en deux versions: une version standard, utilisable sur n'importe quelle machine et une version étendue qui, en utilisant un Extendeur du DOS, fonctionne en mode protégé (à condition de disposer d'au moins un 80286 et d'un méga-octet de RAM).

L'éditeur utilise le jeu de commandes EMACS mais n'est malheureusement pas reconfigurable. Toutes ses fonctions sont cependant accessibles à partir de menus déroulants. Il n'est pas non plus possible de rajouter d'autres programmes au sein de l'environnement, bien que le compilateur et le débogueur s'y intègrent parfaitement.

Le compilateur ZC++ a subi une longue évolution depuis la première version. Le logiciel examiné consiste en fait en trois compilateurs. L'un fonctionne, tout comme l'environnement intégré, en mode réel sur n'importe quelle machine. Un deuxième fait appel à la mémoire virtuelle: il est capable de compiler des programmes de taille plus importante. Le troisième enfin, utilise un Extendeur du DOS pour fonctionner en mode protégé: la taille des programmes compilables est alors quasiment illimitée.

ZC++ est un outil particulièrement complet. Outre les six modèles de mémoire habituels, il offre deux modèles supplémentaires, l'un réservé à la compilation des programmes destinés à être exécutés avec un Extendeur du DOS (donc en mode protégé) et l'autre utilisant la mémoire virtuelle (pour les segments code uniquement), selon un protocole développé par Zortech. Mais le plus gros avantage, c'est certainement la possibilité de générer un code compa-

TURBO C++ Professional de Borland

tible avec les exigences de Windows: en d'autres termes, ZC++ est une alternative qui permet de développer des applications Windows avec tous les avantages d'un langage Objet et sans s'enfermer dans le monopole Microsoft.

Les C++ Tools et la librairie standard offrent aux programmeurs un ensemble de composants logiciels qui peuvent épargner l'acquisition de librairies souvent onéreuses. Ces composants comprennent une bonne partie des structures de données telles que queues, piles, listes, arbres binaires et autres collections que l'on rencontre fréquemment et que, le plus souvent, l'on crée à nouveau à chaque nouveau programme. La librairie comprend des fonctions qui permettent de créer des programmes résidents et d'accéder à la mémoire EMS à l'aide de vecteurs (handles). ZC++ dispose d'un jeu de routines graphiques, dont le code source est également fourni. Cette source comprend les fonctions nécessaires à l'écriture des pilotes de périphériques et il est possible de créer (enfin...) un pilote qui gère, par exemple, une carte Super-VGA en 1024x768 points, 16 couleurs.

La documentation n'est ni au dessus, ni en dessous de la moyenne. La majeure partie de cette documentation est en fait celle de la version précédente Un petit manuel (C++ V2.1 Update Guide) reprend les modifications apportées à la version 2.1 et donne, entre autres, des exemples de programmation pour Windows. Quelques lacunes ont cependant échappé aux correcteurs, notamment lorsque le manuel, qui n'aborde que Windows 2.11, conseille de se référer à un fichier README.DOC pour les renseignements relatifs à Windows 3.0. Le fichier existe effectivement, mais il ne contient rien à propos de l'utilisation du compilateur pour cette version... Il m'a fallu contacter le support technique de Zortech, au demeurant fort courtois et compétent, pour savoir quelles librairies utiliser.

Le plus gros avantage, c'est certainement la possibilité de générer un code compatible avec les exigences de Windows

ZC++ se révèle très agréable à l'usage. Il est au moins aussi rapide que TC++, à condition de ne pas faire usage de l'optimiseur qui est d'une lenteur exaspérante. Dans l'ensemble, ses performances sont correctes. J'ai noté toutefois une fâcheuse tendance à produire plus de code que son concurrent (en moyenne, 4Ko de plus). En revanche, le code produit est quasiment toujours plus rapide (bien que souvent la différence ne tienne qu'à quelques fractions de seconde).

Pourtant, tout n'est pas rose. L'allocation dynamique de mémoire semble avoir quelques difficultés: un programme de compression de fichiers a été incapable de créer ses tampons en modèle "small" et "médium", alors que le même programme compilé avec TC++ y arrive parfaitement. ZC++ dispose d'une option qui permet la génération d'instructions 80286: cette option "plante" systématiquement les programmes Windows.

Enfin, aucune des applications Windows données en exemple ne s'exécute correctement...! Il semble que ce plantage tienne à la façon dont est définie la classe de base, car une application qui ne définit pas de classes pour les fenêtres de Windows est compilée et s'exécute sans problèmes. J'ai été aussi surpris de constater que, bien que ZC++ se réclame compatible avec C++ 2.0, les routines iostreams ne soient pas implémentées.

ZC++ est, dans son ensemble, un produit intéressant, non seulement par ses performances, mais aussi par les "plus" tels que la possibilité d'utiliser un Extendeur du DOS ou la mémoire virtuelle, qui le rendent intéressant dans certains cas particuliers et qui font pencher la balance en sa faveur. De plus, ce compilateur est disponible dans des environnements aussi divers que DOS, OS/2, UNIX et DOS/386. Le code source est naturellement portable sans modifications d'un environnement à l'autre (exception faite des sections de code qui dépendent du matériel sous-jacent ou du système d'exploitation). Les utilitaires qui l'accompagnent (débogueur, éditeur de liens, GREP, etc.) sont faciles d'emploi, fiables et efficaces. Ils n'ont rien de particulier qui pourrait en faire des outils exceptionnels, mais ils font ce que l'on leur demande de faire et ils le font bien. En tout cas, la productivité n'en souffre pas.

A ma droite, le challenger...

Borland a eu l'amabilité de nous communiquer la version "Professional" du compilateur. Cette version comprend, en plus de TC++, le Turbo Debugger 2.0, le Turbo Assembler 2.0 et un produit nouveau, le Turbo Profiler 1.0. Le tout est accompagné de 9 manuels (4 pour le TC++, 3 pour TASM, 1 pour le débogueur et 1 pour le profileur) et de 14 disquettes. L'installation est automatisée et s'est déroulée sans incidents, malgré l'usage d'un disque "exotique" (300 Mo dans une partition unique, SCSI, unité J:).

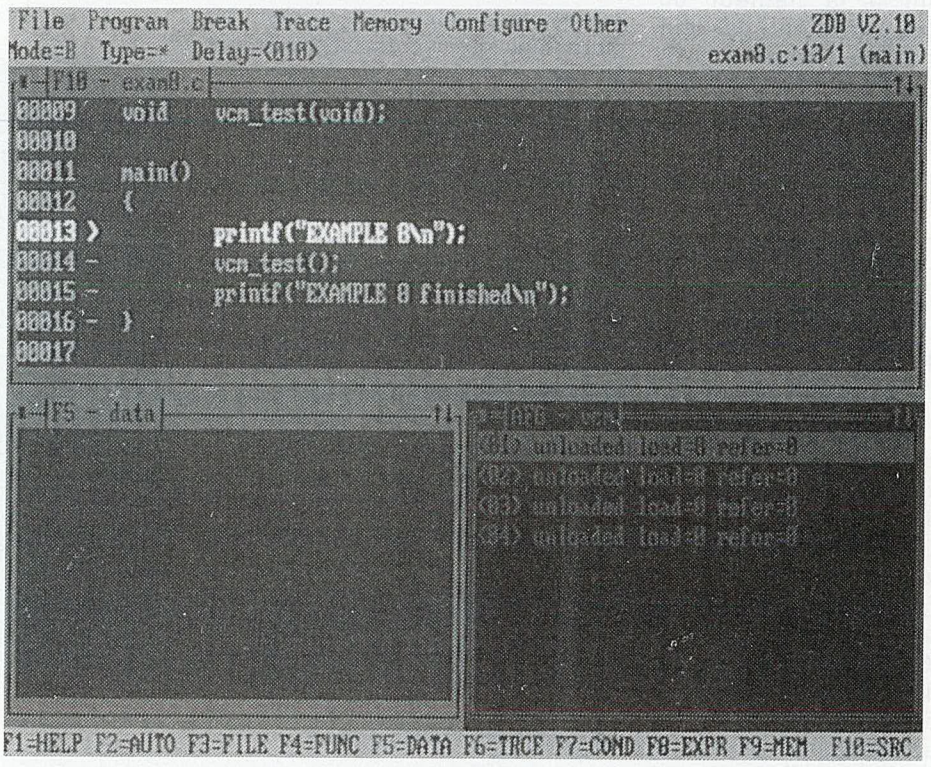
C'est désormais l'usage chez Borland de livrer les compilateurs en deux versions (environnement intégré et version non inter-active) et TC++ n'échappe pas à la règle. L'environnement de développement intégré inaugure une interface utilisateur entièrement repensée, dont l'ergonomie est exemplaire. L'éditeur, point central de cet environnement, a acquis la capacité d'éditer plusieurs fichiers simultanément (jusqu'à 1 Mo par fichier) et d'utiliser la souris. L'affectation des touches reste identique aux autres produits, mais rien n'empêche de l'adapter à ses préférences, d'autant plus que l'éditeur dispose d'un véritable langage de programmation, avec son propre compilateur. L'environnement intégré accepte l'incorporation d'utilitaires externes: ceux-ci apparaissent dans les menus déroulants

Le compilateur est livré avec un certain nombre de classes qui implémentent des types de données de base (listes, piles, nombres complexes, collections triées ou non etc.).Ces classes sont fortement inspirées par SmallTalk, dont elles reprennent par ailleurs les dénominations. Elles sont cependant moins nombreuses que dans ZC++ et ne sont décrites que dans un fichier README. C'est dommage, d'autant plus que cette description est des plus succinctes. Un manuel imprimé, avec des exemples commentés serait bien plus explicite.

Le compilateur est d'une stabilité exemplaire: tous les sources C++ dont je dispose ont été compilés sans problèmes.

TC++ se réclame compatible avec les spécifications C++2.0. Il utilise par défaut les routines iostreams, mais les anciennes routines streams sont également livrées.

Les fonctions graphiques de TC++ s'articulent autour des pilotes de périphériques BGI. La plupart des adaptateurs standard (CGA, EGA, VGA, Hercules, etc.) sont reconnus,



Débogeur

mais les pilotes de cartes Super-VGA font cruellement défaut. A ma connaissance aucun producteur de cartes n'en fournit et la conception d'un tel pilote exige l'acquisition d'un kit de développement auprès de Borland. Il en est de même pour les codes sources de la librairie, qui ne sont disponibles qu'en option (moyennant finances).

La documentation est à la hauteur de ce que l'on avait l'habitude de voir chez Borland et n'appelle pas de commentaires particuliers.

TC++ est entièrement dans la lignée de ses prédécesseurs, aussi bien par la facilité de mise en oeuvre que par la vitesse de compilation. Les exécutables sont le plus souvent plus petits, quoique légèrement plus lents. Le compilateur est d'une stabilité exemplaire: tous les sources C++ dont je dispose ont été compilés sans problèmes.

TC++ est un produit de grande qualité et d'utilisation très agréable. Mais, au moment de faire les comptes, je m'aperçois qu'en fait, il n'a que peu d'avantages sur ZC++. La stabilité exemplaire et la facilité de mise en oeuvre sont ses points forts, mais je reste un peu sur ma faim. En tout cas, il m'est difficile de trouver l'argument qui ferait de TC++ un produit exceptionnel. Les outils qui l'accompagnent sont à la hauteur du compilateur et pourraient, par leur variété et leur qualité, orienter le choix. Il faut noter que TC++ est le seul à être livré (dans la version "Professionnal") avec un assembleur et un profileur.

Et le vainqueur est ...

La question est difficile à trancher. C'est volontairement que je ne donne pas de chiffres: les différences mesurées entre les deux compilateurs, que ce soit à la compilation où lors de l'exécution du code résultant, ne sont pas significatives sur un PC-AT à 8MHz et le seront encore moins sur un 386 ou 486. Il me semble plus intéressant de ne prendre en compte que les caractéristiques des compilateurs, caractéristiques définies en termes de ce qu'ils peuvent ou ne peuvent pas faire.

A mes yeux, la possibilité d'utiliser le C++ pour développer des applications Windows est un avantage immense (mais l'argument est très personnel). Le fait que ZC++ génère systématiquement des exécutables plus rapides est non négligeable. Enfin, disposer, sans frais supplémentaires, du code source des librairies standard est un avantage certain pour qui veut développer des applications pointues. Mais l'environnement intégré fait pencher la balance vers les points négatifs. Un éditeur EMACS, non-reconfigurable, risque d'en mécontenter beaucoup, quoique rien ne vous empêche d'utiliser votre éditeur préféré et de compiler à partir du DOS. Par ailleurs, ZC++ bénéficie d'une certaine ancienneté, qui aurait pu laisser espérer une maturité et donc une stabilité meilleure, mais cette espérance a été partiellement démentie lors des tests.

TurboC++ compte, pour son avantage, une grande stabilité ainsi que l'implémentation des routines iostreams, conformément à la norme 2.0. Toutefois, sachant qu'ATT lui-même n'est pas certain de l'implémentation de cette norme, l'argument perd un peu de son poids. En dehors de ces deux points, TC++ ne peut guère compter que sur les outils qui l'accompagnent pour affronter ZC++. L'incapacité de générer du code compatible avec les exigences de Windows est, par contre, une tare inacceptable, surtout au moment où quasiment tous les développeurs ont fini par se convaincre que Windows est un environnement viable et se décident à écrire des applications. Les outils actuellement disponibles sont trop peu nombreux pour laisser passer une telle lacune.

En somme, si mes besoins de développement ne tenaient pas compte de Windows, mon choix se porterait sur TC++ qui me semble un peu plus stable et plus proche de la norme ATT la plus récente (je reconnais que mon choix est aussi influencé par l'excellent environnement intégré de TC++), tout en sachant que ZC++ satisfait pleinement aux exigences du développement sous DOS. Mais comme les trois quarts de ma production sont centrés sur Windows, ZC++ est un choix nécessaire et il faut reconnaître que les problèmes mentionnés ne gênent en rien son utilisation dans cette optique.

TURBO C++
édité par **BORLAND Incl.**
43, av. de l'Europe
BP. 6
78143 VELIZY VILLACOUBLAY
Cédex

Tél : 39.46.96.69
Fax : 39.46.81.60

TC++ (std, compil seul) 1995 F. HT
TC++ professionnel 2995 F. HT

mis à jour de standard
1.5 et 2.0→ C++ 795 F. HT
US →FR 795

prof. 1.5/2.0 → C++ 1495 F. HT
US→FR 795

ZORTECH C++
importé par **DISTRIC**
123, rue du Château
Techniland 92
92100 Boulogne

Tél : 49.09.10.10
Fax : 46.04.17.19

Prix :
C++ base (compilateur seul)
1495 F. HT

C++ développeur (compil, tools, sources librairie et Debug)
3195 F. HT

PROTECT

modifié par **Bruno VASSEUR**

A la suite de l'excellent article concernant la création d'une protection logicielle, paru dans le numéro de DEBUG Magazine, j'ai voulu , comme beaucoup de lecteurs, je pense saisir le listing des programmes PROTECT.COM et VERIF.COM , dans le but essentiel de comprendre les mécanismes de ce type de protection. C'est au moment de l'assemblage de ces programmes que mes déboires ont commencé.

En effet, PROTECT.COM, notamment sur ma machine, un AMSTRAD PC 1640 avec disque dur 40 Mo et affichage couleur EGA (il est interdit de rire), affichait son message d'accueil tel que défini par le listing, puis rendait la main au DOS sans autre forme de procès et, à fortiori, sans avoir procédé au formatage des pistes 38 et 39 de la disquette à protéger, comme il aurait dû le faire. Heureusement, cependant, le prompt du DOS s'affichait et la machine ne plantait pas, ce qui m'a aidé à comprendre le problème.

J'ai donc tenté de corriger les listings initiaux de manière à faire fonctionner ces programmes et, après avoir galéré d'une manière peu commune, étant totalement ignare en langage machine (un grand merci au passage à MICRO-APPLICATION pour son autoformation à l'assembleur sur PC), j'ai pu faire fonctionner ces routines de protection logicielle comme elles devaient le faire.

Je précise, cependant, ne pas avoir complètement compris ce qui coinçait dans les listings publiés; de ce fait, il est inutile de m'en demander l'explication.

LES MODIFICATIONS

Le but de PROTECT.COM est de créer une protection par formatage non standard des pistes 38 et 39 d'une disquette. Pour cela, il a pour mission de modifier les données contenues dans le Disk Parameter Table ou DPT (en français Table des paramètres disque).

Le programme original, après recherche de l'adresse de la DPT par l'intermédiaire du vecteur d'interruption 1Eh, devrait écrire les nouvelles données de formatage en lieu et place de celles de la DPT originelle, qui étaient préalablement sauvegardées. Constatant que PROTECT.COM ne faisait pas ce pour quoi il avait été créé, je l'ai passé à la moulinette de DEBUG; c'est à ce moment - là que je me suis rendu compte qu'en fait, la DPT ne subissait aucune modification.

J'ai alors transformé le programme de telle sorte qu'au lieu d'écrire les nouvelles données de formatage à la place de celles d'origine, il écrive l'adresse de la nouvelle DPT définie dans le listing à la place de celle de la table originale, contenue à l'adresse du vecteur d'interruption 1 Eh, soit à partir de 0000:0078h. Bien entendu, la modification apportée au programme intègre une sauvegarde de l'adresse de l'ancienne DPT, qui est ensuite réécrite à son emplacement mémoire, en fin de programme, après que celui-ci ait procédé au formatage des pistes 38 et 39 de la disquette contenant l'application à protéger.

Ensuite, PROTECT.COM n'étant rien sans sa routine de vérification associée VERIF.COM, j'ai modifié de la même manière le listing de ce dernier programme, lui faisant permuter les adresses de l'ancienne DPT et de la nouvelle table au lieu de lui faire modifier les données de formatage d'origine comme cela était prévu par le listing publié.

Ce sont ces deux programmes, modifiés et testés, que vous pourrez enfin saisir et faire fonctionner sur vos chers PC, et qui vous aideront, je l'espère, à protéger vos propres créations ou, plus prosaïquement , à comprendre le principe du plombage d'un logiciel.

Le principe de la protection installée reste celui défini par le programme paru dans le numéro 8 de DEBUG Magazine tel que décrit dans le texte qui l'accompagnait. C'est ainsi que toutes les explications apportées par ce texte restent valables, à l'exception de la description des opérations sur la DPT, bien entendu.

De même, le fonctionnement de la routine de vérification associée VERIF.COM reste globalement identique à ce qui était décrit dans l'article n°8 de DEBUG Magazine.Notamment, les codes d'erreur recherchés à la lecture des pistes 38 et 39 sont les codes 02 pour les secteurs 04 et 04 pour les secteurs 09.

Enfin, les considérations finales de ce texte selon lesquelles PROTECT.COM était une démonstration restent valables. En effet, partant du principe qu'un système de protection efficace ne doit être connu que de celui qui le pose, je me suis contenté de faire en sorte que ce programme soit fonctionnel, sans y intégrer de test supplémentaire, ce que je laisse à la discrétion des lecteurs intéressés par le sujet.

CONCLUSION

J'avoue ne toujours pas avoir parfaitement compris toutes les subtilités du plombage d'un logiciel, malgrè le parfait fonctionnement de PROTECT.COM et VERIF.COM. Aussi, je lance ici un appel à la ronde, espérant qu'un prochain article dans cette revue, écrit par quelqu'un plus au fait de ce problème, voudra bien éclairer la lanterne du pauvre ammateur que je suis.

OUF !

UTILISATION DE PARTAGICIELS COMME UN TOUT COHERENT

par Georges Bourne

L'idée première, mais incomplète, que l'on peut se faire d'un partagiciel, est celle d'un logiciel non commercial, issu d'un besoin d'un grand nombre d'utilisateurs, et diffusé soit sous forme gratuite, soit sous forme de contribution volontaire peu coûteuse.

NOTION DE PARTAGICIEL

Lorsqu'une contribution volontaire est demandée, on peut trouver le logiciel soit sous sa forme complète, soit sous une forme d'essai ne possédant pas toutes les fonctionnalités; le versement de la contribution demandée permet alors de recevoir la dernière version complètement fonctionnelle. Dans ces deux cas, on parle alors de partagiciel à contribution volontaire.

L'esprit du "shareware" repose également, dans la mesure du possible, sur la notion de retour à l'auteur, des remarques positives ou négatives sur le logiciel dans le but d'améliorer des versions ultérieures.

Les fichiers sont en général diffusés compactés, et il faut pouvoir disposer des outils de décompactage, eux-même étant des partagiciels.

Pour être concret, je propose de présenter ci-dessous ceux que j'utilise, d'expliquer comment j'ai pu les obtenir, et dans le cas de contribution volontaire, comment celle-ci a été réglée.

ENVIRONNEMENT MATERIEL ET SON USAGE

J'utilise un micro-ordinateur compatible XT, 2 lecteurs de disquettes 360Ko, un disque dur 20 Mo, un écran couleur EGA, 640 Ko de mémoire vive, microprocesseur 8086, coprocesseur 8087, un modem V21 V22 V23 V25 V25b.

L'usage principal du micro concerne :

* programmation en Turbo Pascal ;
* traitement de texte pour courrier et rapports ;
* tableur pour comptabilité ;
* télématique (minitel et serveurs) ;
* jeux (dans une moindre mesure).

■

PARTAGICIELS UTILISES ET LEURS LOCALISATIONS

```
A LA RACINE C:\

ANSI    COM          3072    31/08/90    9:27
        1 fichier

REPERTOIRE  C:\DOS\UTIL\

.          <DIR>              24/08/90   18:50
..         <DIR>              24/08/90   18:50
AMENU COM              6820   18/06/90   23:35
DSZ   COM             51628   26/08/90   20:04
NOVIRUS EXE           10672   10/03/88   21:07
OPCOLOR EXE            6609   28/07/88   10:31
OPTUNE  EXE           47889   28/07/88   10:55
PKUNZIP EXE           22540   15/03/90    1:10
PKZIP   EXE           32880   15/03/90    1:10
QCONFIG DAT            7348   27/06/90    2:10
QCONFIG EXE           22143   27/06/90    2:10
QDELCOM                5814   22/07/90   10:04
QEDIT EXE             46582   12/09/90   22:40
QHELP TXT              1953   27/06/90    2:10
LIFE  COM             10240   20/05/89   10:16
VACINE EXE            23024    3/03/88   21:51

        16 File (s)
```


[SERVICE LECTEURS : INTERESSANT = 34 / MOYEN = 35 / MAUVAIS = 36]

```
REPERTOIRE C:\MODEM\EZ133\

.          <DIR>              27/09/90    20:23
..         <DIR>              27/09/90    20:23
QMAIL      <DIR>              27/09/90    20:23
SAUVE      <DIR>              27/09/90    20:23
ADDRESS BK          0         27/09/90    21:45
CONFIG  EXE     64966         23/02/90    10:53
EZ-RDR  CFG      6777         17/11/90     7:54
EZ-RDR  EXE    151696          7/09/90    14:30
        8 File(s)

SUR DISQUETTE VERROUILLEE SCAN

        Directory of  A:\

SCAN    EXE     43277         24/04/90    14:24
SCANV62 DOC     30626         25/04/90     8:45
VALIDATE COM     6485         31/10/89    16:16
VALIDATE DOC     1804         18/03/90    17:38
VIRLIST TXT      8727         24/04/90    14:04
        5 Files (s)

SUR DISQUETTE VERROUILLEE CLEAN-UP

        Directory of  A:\

CLEANP59 ZIP    45056         18/03/90    18:08
CLEAN   EXE     54959          4/03/90    14:20
VIRLIST TXT      7712          4/03/90    12:45
VALIDATE  COM    6485         31/10/89    16:16
VALIDATE  DOC    1653         28/10/89    22:13
REGISTER DOC     2610         15/01/90    10:13
CLEAN59  DOC    11281          4/03/90    17:22
        7 File(s)
```

UTILISATION FONDAMENTALE A CHAQUE BOOT DE L'ORDINATEUR

Pour introduire cela, le plus parlant consiste à jeter un coup d'oeil sur les fichiers CONFIG.SYS et AUTOEXEC.BAT :

CONFIG.SYS :

```
country 33
buffers 20
files 25
shell command.com /p /E:512
```

On remarquera l'accroissement de la taille mémoire réservée à l'environnement porté à 512 octets pour les besoins des partagiciels utilisés. Le débutant a souvent des problèmes avec cet environnement pour cause de débordement liée aux "PATH" et aux "SET".

Ainsi, ce ne sont pas moins de 5 partagiciels qui sont lancés à chaque boot de la machine. De plus, comme cela a été vu, l'essentiel des partagiciels étant dans le répertoire C:\DOS\UTIL\ un PATH est nécessaire pour les appeler de partout. Enfin des variables d'environnement sont définies pour utiliser le protocole de transfert ZMODEM et pour le "READER EZ-RDR off line" de messageries des serveurs. Passons en revue ces partagiciels :

LE PILOTE VIDEO : ANSI.COM

```
AUTOEXEC.BAT :
        echo off
        verify on
ainsi                                   (1)
        prompt $e[41m$e[37m$p$e[m$e[32m$g$e[40m$e[36m
        path c:\dos;c:\dos\util;c:\turbo;c:\qpro;
SET DSZLOG=C:\MODEM\PCPLUS\DSZ.LOG
SET DIRRX  =C:\MODEM\PCPLUS\RECOIFIC\
SET EZRDR=C:\MODEM\EZ133\
keybfr
optunec:/ON V                           (2)
gr_ega
mode COM2:96,N,8,1
novirus                                 (3)
pause
vacine c                                (4)
life /A /F                              (5)
pause
cls
```

ORIGINE : PC MAGAZINE américain disponible sur le serveur SUPTEL, coût GRATUIT.Il remplace avantageusement le DRIVER ANSI.SYS du DOS, qui est appelé dans CONFIG.SYS par la commande " DEVICE = ANSI.SYS" ; il permet des affichages ANSI plus rapides et comporte plusieurs options de configuration, ici sans paramètres passés, c'est la configuration par défaut qui est utilisée. Il détecte si ANSI.SYS est déjà installé.

Note : Il faut le mettre à la racine, car il est appelé et installé avant "PROMPT" et "PATH", ou indiquer son chemin d'accès dans son appel par l'AUTOEXEC.BAT.

INSTALLATION : immédiate.

INCOMPATIBILITES : aucune.

MENTION : TRES BIEN.

LE GESTIONNAIRE DE DISQUE : OPTUNE

ORIGINE : Gazelle Systems, Provo, UTHAH 84601, USA, disponible sur le serveur RCI Calvacom dans la BIB de la CITE OUF, coût GRATUIT.Superbe gestionnaire des disques, en particulier du disque dur. Lors de chaque boot, activé par la commande "optune c: /ON ", il vérifie s'il ne trouve pas des fichiers ouverts et non fermés, puis effectue la défragmentation de tous les fichiers. Cela ne prend que quelques secondes. Une fois par semaine, utiliser OPTUNE pour faire un réalignement des fichiers, c'est-à-dire par répertoire les rendre contigus, non fragmentés et classés par ordre alphabétique (autres options possibles). Tous les 6 mois ou une fois par an, effectuer une vérification de l'écriture/lecture (plusieurs options) et aussi un formatage du disque de bas niveau qui ne détruit pas les données écrites. D'autre opérations de gestion du disque sont possibles, les disquet-

tes peuvent également être gérées. La documentation en anglais est complète, elle donne toutes les explications nécessaires.
INSTALLATION : immédiate.
INCOMPATIBILITES : n'aime pas les programmes résidents, donc à lancer dès que possible au début de l'AUTOEXEC.BAT ou déloger les résidents ; n'aime pas être appelé après "PRINT" du DOS ; ne supporte pas le disque-cache. Ne fonctionne pas avec certains disques durs exotiques.
MENTION : TRES, TRES BIEN, et même plus car son utilisation régulière, assure la sécurité du stockage sur disque des données, et accroît la durée de vie du disque dur, celui-ci travaillant dans de meilleures conditions.

VERIFICATION DE L'INTEGRALITE DES FICHIERS DE COMMANDE DU DOS : NOVIRUS

ORIGINE : Matt Hill , Cherry Hill, NJ 08034 USA, disponible sur le serveur RCI Calvacom, BIB de la CITE IBM ; coût $5, versement effectué par l'intermédiaire de SOLIDASOFT - 66 boulevard Exelmans - 75016 PARIS.Il vérifie par CRC les fichiers de commande DOS et signale les anomalies par une petite sirène et message à l'écran. La documentation en anglais se comprend sans problème.
INSTALLATION : lors du premier lancement, un fichier est crée pour recevoir les éléments calculés, et qui servent ensuite de référence à chaque boot.
INCOMPATIBILITE : aucune, mais attention, ce n'est pas un détecteur de virus, il signale simplement que le DOS n'a pas été corrompu.
MENTION : TRES BIEN, et très rassurant lors du début du travail sur le micro.

VERIFICATION DE L'INTEGRALITE DES FICHIERS DE COMMANDE DU DOS : VACINE

ORIGINE : Art Hill, La Grange, IL 60525 USA, disponible sur le serveur RCI Calvacom dans la BIB de la CITE IBM ; coût laissé à l'appréciation de chacun versé par l'intermédiaire de SOLIDARSOFT. Vacine vérifie la l'integrité des fichiers de commande du DOS, selon d'autres critères que NOVIRUS.
INSTALLATION : elle se fait sans difficultés, il suffit de lancer VACINE et une page d'aide apparaît.
INCOMPATIBILITE : aucune.
MENTION : TRES BIEN.

PROTECTION DU DISQUE DUR : LIFEGARD

ORIGINE : auteur : Jean-Louis VILLERME, disponible chez Logiciel et Médias 125 rue de Saussure, 75017 PARIS ; coût : environ 300 F, 20 % de cette somme étant reversée à l'Association Enfance et Partage. Le manuel, en français, donne toutes les explications pour protéger effectivement son ordinateur, plusieurs options sont possibles, l'une d'elle interdit toute écriture sur le disque dur: idéal pour tester des disquettes suspectes. L'option de base en particulier empêche le formatage du disque dur et préserve les secteurs de boot et les partitions. Désinstallation possible. Dans l'AUTOEXEC.BAT lancé avec les options : "LIFE /A /F" , LIFEGUARD interdit le formatage et l'écriture par adresses absolues sur le disque dur, protège la FAT, et, étend cette protection aux disquettes. A toujours lancer tout à la fin de l'AUTOEXEC.BAT. LifeGuard joue à la fois le rôle d'anti-virus et le rôle d'anti-fausses manoeuvres.
INCOMPATIBILITE : aucune, mais n'aime pas "PRINT" du DOS. Il faut faire attention à ne pas lancer LIFEGUARD plusieurs fois à la suite par inadvertance, cela peut occasionner des anomalies bizarres, il suffit alors de relancer le boot. **Attention :** le désinstaller obligatoirement avant d'utiliser OPTUNE.
MENTION : TRES, TRES BIEN et logiciel indispensable.

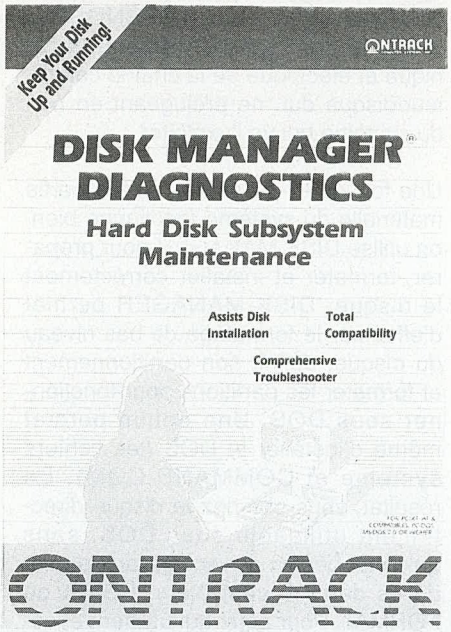
UN EDITEUR DE TEXTE PERFORMANT : QEDIT

ORIGINE : SemWare, Mariette, GA 30062-5003 USA, version d'essai disponible sous forme de disquette (60F) diffusée par OUF : Ordinateurs Utilisateurs France, 27 rue des Bleuets, 75011 PARIS.
Enregistrement : $44, on reçoit alors directement la version enregistrée sous 10 jours, le paiement le plus pratique est par carte VISA internationale ou chèque libellé en dollars, il suffit pour cela de remplir le bulletin que l'on trouve avec la version d'essai. Pour $55, la disquette est accompagnée d'un manuel imprimé contenant la documentation. La version à $44, plus économique, contient la documentation sous forme d'un fichier texte à lister sur son imprimante. C'est un éditeur de texte très performant, qui fait tout et même des choses inattendues telles que lorsque l'on tape "(" les 2 parenthèses sont mises en place l'écriture se faisant à l'intérieur, de même pour "". Il faut disposer des quatre fichiers : QCONFIG.DAT, QCONFIG.EXE, QEDIT.EXE (Q.EXE sur la disquette) et QHELP.TXT. La configuration se fait sans problème en lançant "QCONFIG". Les options sont nombreuses et permettent une parfaite configuration, d'une part à son matériel et d'autre part à l'utilisation que l'on veut en faire. Les options par défaut font que, de toute façon, sans ré-configurer, cela marche déjà. Possibilité de faire un joli choix de couleurs. Pour utiliser Qedit, seul le fichier QEDIT.EXE est nécessaire. La documentation, très fournie et complète, dit tout ce que l'on peut faire et comment. Et curieusement on retrouve pratiquement toutes les commandes de l'éditeur du TURBO PASCAL, TURBO BASIC ou de WORD, si bien que sans se poser de question, si l'on a déjà utilisé ces éditeurs on est opérationnel immédiatement. Pour lancer : "QEDIT <filename>" ENTER ou "QEDIT" ENTER. Les déplacements dans les répertoires sont possibles, et se font par menu déroulant. Pour appeler les fonctions de l'éditeur, on peut utiliser directement les combinaisons de touches, ou bien en validant "ESC" faire apparaître en haut de l'écran la barre du menu puis les fenêtres des sous-menus. Macros instructions possibles, tracés très commodes de cadres et très nombreuses autres possibilités.
INCOMPATIBILITE : aucune, au contraire facilement intégrable dans certains logiciels réclamant un éditeur de textes.
MENTION : TRES TRES BIEN, il n'est pas compréhensible que QEDIT ne soit pas livré avec tous les micros vendus. Ce présent texte a été préparé avec QEDIT.

.....

... La suite au prochain numéro.

PODIUM



DES OUTILS INDISPENSABLES

Il y a des logiciels d'agrément, indispensables ou superflus, visibles et invisibles, ceux qui sont installés dans une machine sans que personne ne s'en rende compte et ceux qui brillent quotidiennement de toutes leurs couleurs sur tous les écrans. Certains sont interchangeables, certains sont uniques.

Dans cette nouvelle rubrique, nous allons vous présenter, des matériels ou logiciels que nous avons particulièrement aimé, pour leur utilité, originalité ou qualité. Si vous appréciez particulièrement un produit, soumettez-nous sa candidature. Les bons produits méritent d' être connus!

Dans la mallette d'un technicien, DISK DIAGNOSTIC et DISK MANAGER d'ONTRACK sont vitaux, indispensables, inévitables, irremplaçables. Un homme-système, un technicien rompu aux problèmes de terrain, proche des utilisateurs, ne peut pas s'en passer.

Les logiciels d'ONTRACK s'occupent des disques durs, le périphérique le plus important, le plus sensible et le plus fragile de l'ordinateur. Comme tous les utilitaires, les logiciels ONTRACK comblent la lacune laissée par le DOS d'IBM, qui ne fournit aucun outil utilisable pour manipuler le sous-ensemble disque dur au niveau matériel.

Tout d'abord, il y a DIAGNOSTIC : un logiciel qui permet de tester extensivement l'ensemble contrôleur/disque d'un PC/XT/AT. Prenons un exemple type : un jour, vous êtes appelé au secours d'un ordinateur dont le disque dur est « planté ». Vous ne savez pas si cela vient du contrôleur, du câble ou du

disque dur lui-même. Une démarche voltamètre/oscilloscope ne serait d'aucun secours, démesurée par dessus tout. Ou, dans une autre situation, vous devez installer un disque dur lamba dans un « compatible » quelconque, dont vous ne connaissez que la marque et le type du contrôleur et du disque dur. Evidemment, au premier égard, rien ne marche.

Vous démarrez avec un DOS « propre », sans CONFIG.SYS ni AUTOEXEC.BAT, vous lancez DIAGNOSTIC. Tout d'abord, le logiciel vous permet de définir combien de disques durs sont connectés au système, de définir leur type en choisissant dans la liste des modèles reconnus. Dépendant de la version, vous trouvez quelques dizaines de modèles choisis parmi quelques 20 à 30 fabricants connus. Il y a peu de chance pour que votre disque dur ne soit pas répertorié, d'ailleurs, si c'est le cas, le programme vous permet de saisir les paramètres de votre modèle particulier « à la main ».

Une fois les données du disque définies, un menu vous propose six tests : contrôleur, positionnement de têtes, lecture aléatoire, lecture/écriture aléatoire et écriture/lecture ECC (Error Correction Code - test de capacité de correction du contrôleur, sur un cylindre spécifié). Le sixième test vérifie l'état de la surface. Les tests peuvent s'enchaîner automatiquement ou peuvent être lancés un par un. Dans le cas d'un problème, un message en clair indique la source de l'erreur et suggère des remèdes éventuels. Tous les tests sont vitaux et testent progressivement la chaîne contrôleur-disque. En effet, on ne peut passer aux tests de la mécanique sans que le contrôleur ne fonctionne bien ou que le câble ne soit correctement branché! La lecture/écriture serait impossible sans que la mécanique de positionnement des têtes ne soit correcte. Le test ECC lui-même constitue l'ultime démarche, qui teste l'aptitude du contrôleur à corriger un train de bits de longueur allant de 0 à 11 bits. Les tests sont rapides, toute la démarche (sur un bon disque) ne prend plus que quelques minutes. Ne vous faîtes pas d' illusions - si l'un des tests échoue, vous n'avez aucune chance de pouvoir utiliser le système. Changez le contrôleur, vérifiez le câblage, vérifiez la configuration du disque (cavaliers de configuration, résistance de fin de ligne), remplacez le disque. Uniquement une suite impeccable d'exécution vous permet de passer à l'étape suivante : installer le disque dans le système.

Il est à souligner que DIAGNOSTIC est utilisable avec les systèmes prévus pour fonctionner aussi sous d'autres systèmes que le PC DOS : UNIX, Novell etc. En effet, DIAGNOSTIC vérifie le bon fonctionnement mécanique et électrique de la chaîne contrôleur/disque dur, ne préjugeant en rien du système qui va l'exploiter.

Une fois que l' on est sûr que la partie matérielle du système fonctionne bien, on utilise DISK MANAGER pour préparer, formater et installer correctement le disque. DISK MANAGER permet d'effectuer le formatage de bas niveau du disque, définir son partitionnement et formater les partitions pour fonctionner sous DOS. Une option permet même d'installer le DOS (les fichiers système et COMMAND.COM). En résultat, vous obtenez le disque directement utilisable sous DOS, sans qu'on ait besoin de recourir aux procédures dans le contrôleur, FDISK ou FORMAT. Pour ceux qui utilisent encore le DOS version 3, DISK MANAGER permet aussi d'utiliser les partitions de la taille supérieure à 32Mo.

L'utilisation du DISK MANAGER commence par la configuration de la CMOS de l'ordinateur. Comme dans DIAGNOSTIC, vous choisissez dans une table la marque et le type du disque installé dans le système. Le type du disque dans la CMOS est ajusté à la valeur qui correspond le mieux au disque concrètement installé. En effet, il arrive souvent que dans la table de disques (stockée dans la ROM de l'ordinateur) on ne peut pas trouver le type qui correspond exactement au nombre de têtes et cylindres de votre disque. Et pourtant, tous les outils du DOS (notamment FDISK ou SMARTDRIVE de Windows) et DRDOS, se réfèrent aveuglement à cette table. Ainsi, sans DISK MANAGER, vous perdez une partie de la surface du disque, inaccessible à cause du manque d' information transmise de la ROM à FDISK. DISK MANAGER utilise ses propres paramètres, sans se référer à la ROM. Après l'installation, au démarrage du DOS, un pilote se charge à partir du CONFIG.SYS, suppléant aux informations fournies par la ROM et permettant d'accéder toute la surface du disque.

L'utilisation du DISK MANAGER est facile - prenons un exemple. Au démarrage, choisissons l'utilisation manuelle (en spécifiant le paramètre du démarrage : DM (automatique) ou DM /M, (M voulant dire manuel)). Dans un menu qui va apparaître choisissez les étapes successives de l' installation du disque : configuration de la machine, préparation, partitionnement. DISK MANAGER détecte le contrôleur, permet de définir le nombre des disques connectés au système, définir leur marque (fabricant) et type. Une fois que les paramètres des disques sont définis, vous passez à l'étape de formatage du bas niveau. DISK MANAGER vous permet de saisir l'emplacement de mauvais secteurs, tels qu'ils sont indiqués par le fabricant lors de tests à l'usine. Je vous recommande d'ignorer ces informations et présumer que le disque est parfait : vous arriverez souvent à récupérer quelques zones présumées mauvaises, qui arrivent à se réformater après plusieurs passages. En effet, parfois une mauvaise zone a pour origine une imperfection temporaire de positionnement de têtes ou une impureté déposée à un endroit - une fois le disque « rodé », la zone est parfaitement utilisable, sa surface magnétique étant bonne. Lors du formatage, on définit le facteur d'interlacement : dans la plupart des cas, dans les AT classiques elle doit être de 2, ce facteur peut changer dans le cas d'autres contrôleurs ou paramètres du CPU (voir aussi NOTE 1).

Une fois le disque formaté et vérifié, il est prêt à être utilisé - il doit être partitionné. Dépendant du système utilisé (DOS, DR DOS etc.), vous pouvez pour cela continuer à utiliser DISK MANAGER ou utiliser le logiciel fourni avec votre système opérationnel (par exemple, FDISK du DOS et DR DOS). Dans le cas des anciennes versions du DOS (version 3 ou moins), DISK MANAGER est plus performant, vous permettant de définir plusieurs partitions, de taille supérieure à 32Mo (sauf celle de démarrage). Un autre menu vous permet de définir et formater chaque partition, en installant même le système sur la partition de démarrage. Encore une fois, le processus est rapide et l'utilisation facile - au bout de quelques minutes votre disque sera préparé pour être utilisé, sans problème, ni accroche.

En conclusion, ONTRACK DIAGNOSTIC et DISK MANAGER sont irremplaçables et absolument nécessaires dans votre mallette du technicien. Hautement recommandés!

NOTE 1

FACTEUR D'INTERLACEMENT

Le facteur d'interlacement dépend du type de contrôleur et de la vitesse de CPU. Il est difficile de définir d'emblée la valeur optimale (assurant le plus haut taux de transfert), particulièrement dans le cas des systèmes assemblés de pièces venant de plusieurs sources, non prévus par le fabricant de la carte mère. Il existe de nombreux outils pour définir la valeur optimale (taux de transfert le plus élève). Notons HTEST/HDISK de Kolod Research (édité par MACE), la version 5 des Utilitaires Norton ou DosUTILS d'ONTRACK.

Il est hautement recommandé d'exécuter au moins une fois un de ces utilitaires sur votre système : à plusieurs reprises, dans des systèmes divers, même « neufs », il m'est arrivé de constater la valeur incorrecte, dégradante des performances deux ou trois fois par rapport aux performances possibles!

NOTE 2

LOGICIELS ONTRACK

ONTRACK Inc. s'est spécialisé depuis longtemps dans le domaine de préparation, utilisation et maintenance des disques durs. A part de DIAGNOSTIC et DISK MANAGER pour le PC DOS, la gamme de logiciels d'ONTRACK englobe aussi un jeu des utilitaires similaires pour Netware de NOVELL. DISK MANAGER/N permet de préparer les disques durs pour être utilisés sous Netware et remplace avantageusement COMPSURF de Novell, en utilisant toute la surface disponible sur disque. DISK MANAGER/N permet notamment d'installer des disques non reconnus par NETWARE, dont les paramètres excédent les tables dans la ROM de l'ordinateur. NETUTILS sont d'une valeur vitale sous NETWARE lorsque il s'agit de récupérer les disques endommagés, fichiers déletés ou perdus. C'est souvent l'ultime recours pour récupérer un par un les fichiers d'un serveur « planté », qui ne veut plus démarrer sous NETWARE. Il faut le vivre une fois pour l'apprécier!

Il existe aussi une version spéciale de tous ces utilitaires pour PS/2 d'IBM. Les utilisateurs de MAC ne sont pas oubliés non plus, ils peuvent utiliser le logiciel DISK MANAGER Mac, apportant les mêmes services que DISK MANAGER pour PC.

DISK MANAGER est connu par le large public grâce à ses versions « bridées », limitées à un fabricant. En effet, ONTRACK vend souvent ce logiciel à un ou l'autre fabricant de disques durs, pour aider ses utilisateurs à installer les disques durs dans des divers systèmes « exotiques ». Ces versions rendent les mêmes services et fonctionnent de même manière que la version « universelle », sauf qu'on ne peut pas l'utiliser pour les disques d'une autre marque et on ne peut modifier ses paramètres non plus.

Enfin, la société ONTRACK Inc. possède aussi une division spécialisée dans la maintenance des disques durs qui assure la réparation et récupération des données sur des disques en panne.

SPEEDSTOR - LA CONCURRENCE

Il existe aussi un autre logiciel ayant les mêmes finalités que DISK MANAGER d'ONTRACK : SpeedSTOR de Storage Dimensions (MAXTOR Company). L'ensemble, englobant l'utilitaire de DIAGNOSTIC et l'utilitaire de préparation, est également connu sous forme « bridée » ou universelle. SpeedSTOR, plus « technique » dans son approche, permet notamment de manipuler directement les partitions d'autres systèmes opérationnels que le PC DOS.

ONTRACK COMPUTER SYSTEMS Inc.

ONTRACK DATA RECOVERY Inc.
6321 Bury Drive
Suites 15-19
Eden Praire, Minnesota 55346
USA
tel : (612) 937 1107
fax : (612) 937 5815

Prix :

DISK MANAGER : 124.95 $
DIAGNOSTICS : 49.95 $
DosUTILS : 99.95 $
DISK MANAGER MAC : 69.95 $
DISK MANAGER /N : 249.95 $
NETUTILS : 99.95 $

DISQUETTE LISTING DEBUG N° 10

1. KERMIT pour OS/2
(article page 48)

2. PROTECT & VERIF
nouvelle version
(article page 65)

Pour obtenir la disquette DEBUG 10, nous retourner le coupon détachable incarté au centre.

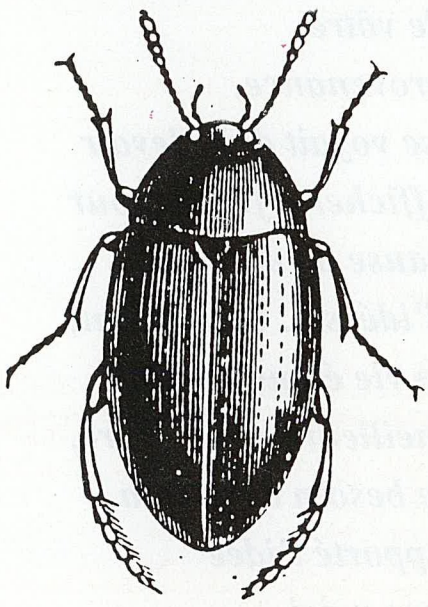
DISQUETTES UTILITAIRES DEBUG MAGAZINE

Utilitaires de compression et d'archivage, chaque disquette est vendue au prix de 47 Frs TTC

DISQUETTE PKPAK
PCSIG n° 1330

DISQUETTE LHARC
PCSIG n° 2018

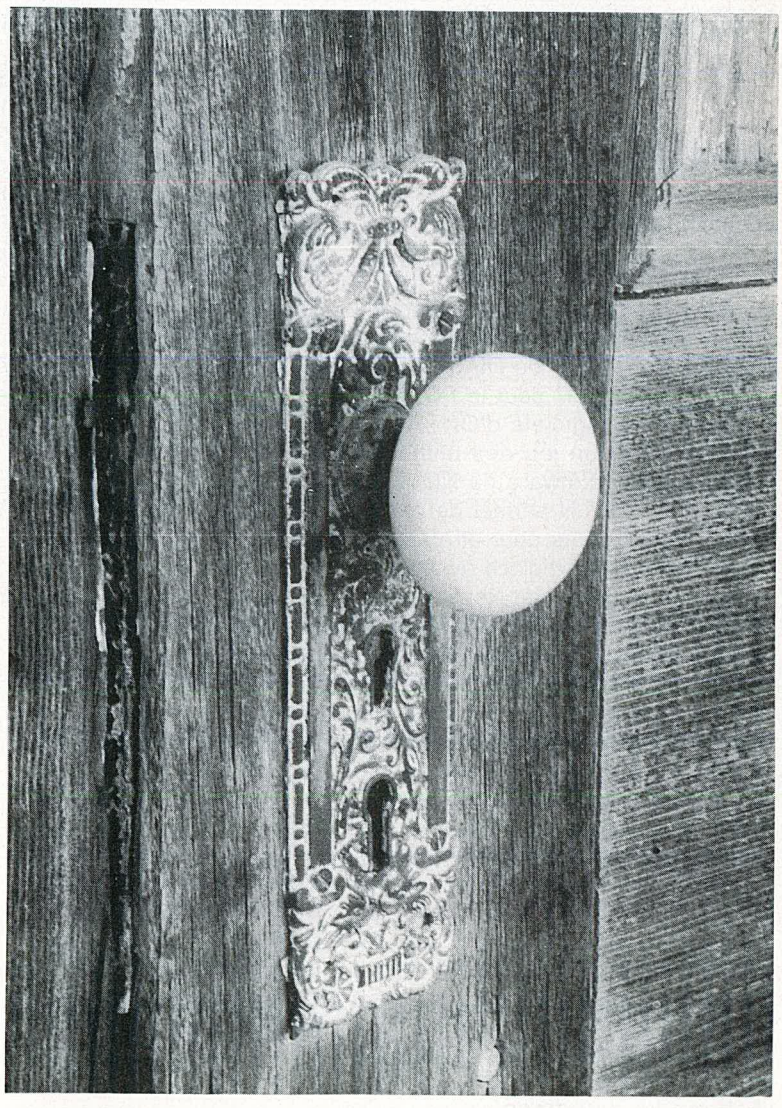
DISQUETTE LZE, LZESHELL
PCSIG n° 2316



IDEE DU JOUR : SENTINELLE AUTOMATIQUE

par *ART*

Le bouclage du numéro approchait. Votre fidèle rédacteur, dépourvu de toute inspiration tant de son cru que de vôtre provenance, se voyait déjà devoir afficher « fermé pour cause de manque d'idées ». Cependant, la vie étant la meilleure conseillère, le besoin du jour a apporté l'idée que voici.



[SERVICE LECTEURS : INTERESSANT = 40 / MOYEN = 41 / MAUVAIS = 42]

La sécurité micro-informatique devient un sujet de plus en plus brûlant. Il n'y a pas de quoi s'étonner, les micro-ordinateurs occupent les places de plus en plus prépondérantes dans notre vie professionnelle. Nous leur confions de plus en plus des données importantes ou confidentielles, en conséquence, les problèmes de protection ou de contrôle d'accès sont de plus en plus compliqués.

Cet aspect de la micro-informatique devient particulièrement complexe et d'actualité dans l'environnement des Réseaux Locaux. Dans ces systèmes, plusieurs utilisateurs partagent des ressources matérielles et logiciels, exploitent les mêmes fichiers, échangent des informations, accèdent aux mêmes données. A l'image de l'Entreprise, Administration, Atelier de Recherche ou de Fabrication, le système d'accès et la capacité à manipuler les informations est généralement bien définie, dépendante de la place occupée par l'utilisateur (personne physique) dans la hiérarchie de l'organisation. Généralement, dans l'espace qu'il lui est alloué, chacun peut faire ce qu'il veut, à la condition que ses données soient communicables à certains autres utilisateurs du Réseau. Les groupes de travail accèdent ensemble à certaines ressources (bases de données, imprimantes etc.) bien définies, d'autres encore ont leurs propres ressources administrées à part.

Des systèmes opérationnels des Réseaux Locaux dignes de ce nom prennent toutes ces tâches en charge automatiquement, c'est d'ailleurs leur rôle le plus important. L'exemple : Netware de la société Novell. Selon l'ordre et la hiérarchie décidée par l'administrateur du système, Netware contrôle l'accès à des fichiers et répertoires, autorise ou non l'usage de tel ou tel périphérique du système, dans des limites horaires et géographiques parfaitement contrôlables et définissables. L'administrateur peut décider qui va utiliser quelle donnée, à partir de quel poste et quand, car même les plages horaires d'accès au système sont cloisonnables. On peut décider, par exemple, que l'ensemble de paie ou la comptabilité de l'Entreprise peuvent être utilisés qu'à partir du poste du comptable, entre 9h du matin et 18h du soir, par l'utilisateur « COMPTA » seul. Ce système garanti une très grande sécurité et confidentialité aux informations confiées au système. Il y a cependant une lacune, une brèche qu'il serait bon de colmater. Sous Netware, pour se connecter au système, l'utilisateur (une entité virtuelle définie par ses droits et son nom) doit s''identifier en indiquant son nom et son mot de passe particulier. Ce processus de connexion, appelé « login », est exécuté grâce à un programme spécial fourni avec Netware, LOGIN.EXE. L'utilisateur « existe » dans le système tant qu'il ne se déconnecte pas par une autre procédure, appelée « logout » (programme LOGOUT.EXE). Pendant que sa connexion est valide, toute personne physique peut exploiter à son gré la connexion, dans les limites des droits qui lui sont assignés. Par exemple, en simplifiant, une personne entrée accidentellement dans le bureau du comptable, en son absence peut accéder à la comptabilité de l'Entreprise, si ce dernier a laissé son ordinateur en marche sans se déconnecter avant de partir.

Dans cette nouvelle rubrique, intitulée « IDEE », nous allons publier les idées diverses sur les programmes ou dispositifs à réaliser. Des idées géniales, imprévisibles, nouvelles, créatives, farfelues (qui sait?) en somme, la diversité des idées à réaliser.

En effet, DEBUG MAGAZINE manque (comme tous les autres, par le temps qui court) de pétrole, mais pas d' idées. Chaque programmeur se pose quotidiennement la question : que faire lorsque je terminerai le programme en cours ? Ce programme n'est jamais terminé, mais à des moments rompus, on commence une nouvelle réalisation...

Tous, nous avons aussi des besoins. Tous les jours, nous nous disons : « ah, si cela existait, je serai moins ennuyé »!

Décrivez-nous vos besoins, envoyez les idées, même celles qui semblent le plus bizarre. Qui sait ? Peut-être allez vous déclencher une découverte révolutionnaire ?

Deux cas peuvent se présenter pour expliquer le défaut de la déconnection: soit c'est un oubli, soit c'est une nécessité technique. L'utilisateur physique peut bien quitter son poste de travail bien que la tâche informatique sur ses données n'est pas encore terminée (une longue reindexation d'une base de données, recalcul d'un grand tableur, relance sur comptes clients, etc.). En absence des doigts de l'utilisateur lui-même, il n'existe aucun moyen pour déconnecter automatiquement une station, une fois la tâche terminée.

C'est précisément l'objet de notre idée du mois : créer un programme (de préférence un TSR, résident en mémoire de chacune station de travail) qui permettrait de déconnecter automatiquement (exécuter un « logout ») la station donnée au bout d'un certain temps de son inactivité. Les critères de déconnection pris en compte peuvent être : soit la fin de fonctionnement d'un programme, soit l'inactivité écran/clavier du poste, comme dans ces « économiseurs » d'écrans qui éteignent l'écran au bout d'un certain temps de l'inactivité.

Le critère de l'activité clavier/écran est facile à analyser et prendre en compte par le programme : il suffit pour cela de tester des interruptions adéquates. Le fonctionnement du programme est plus délicat : par exemple, dans le cas de recalcul d'un tableur ou relance clients, bien que la fonction (tâche) lancée est arrivée à la fin, le programme lui-même n'est pas pour autant terminé - on ne retourne pas au DOS. Notre TSR de sécurité devrait l'interpréter correctement et pouvoir déconnecter la station une fois tous les calculs terminés, de « l'intérieur » du tableur ou du programme de comptabilité même.

Remarquons qu'une fonctionnalité semblable existe dans des réseaux larges, comme TRANSPAC : au bout d'un certain temps sans aucune activité (flux de données), le réseau libère la connexion et ferme tous les circuits attachés.

Sous Netware, un tel programme, appelé « SENTINELLE AUTOMATIQUE », préserverait mieux la confidentialité de l'accès aux données de l'utilisateur. SENTINELLE AUTOMATIQUE préviendrait tous les yeux et les doigts trop curieux (ou malveillants) des intrusions indues, provoquées par l'oubli ou une contrainte technique.

PETITES ANNONCES LECTEURS

ECRANEUR: logiciel de création, gestion et utilisation de fenêtres, menus déroulants, etc...utilisable très simplement sous BASIC, assembleur, C/690 Frs. FINGERHUT, 22 rue Kléber - 95100 ARGENTEUIL.

Vous avez des besoins en informatique (formation, conseil, développement d'application...). Alors n'hésitez pas à me contacter.
Tél : 30 38 15 04

Carte FAX ATFAX 9600 Groupe 3 Logiciel gérant la souris, annuaire, conversion de fichier TIAA. Editeur graphique... Votre FAX devient un scanner. Liaison PC:AC ADR.
Mr GONZALEZ, 2 Chemin Tuilerie, 33100 BORDEAUX.

Portable Toshiba 3100E. Disque dur 20 Mo. Ram 1Mo. Ecran Plasma. DOS 3.3. Housse de transport et nombreux logiciels. 16.000F. à débattre.
Tél: (1)46.77.14.43

Imprimante 1101 pour COMMODORE PLUS 4.
Tél: 86.58.42.77

A vendre **Ordinateur SANYO. Modèle 5.5.5.**
900 F.
S'adresser au journal.

CLUB GMT diffuse Sharewares plus exclusivité - DOC. DISK 5.25 C/4 Timbres - pour Compatible IBM - PC
28 rue du Rendez-Vous - 75012 Paris

Vends **Imprimante laser HP** Laserjet ii HP, 2 Mo de mémoire, cartouche F. 8 700Frs. Laisser message au journal, REF 9/5

Ordinateur AMSTRAD PC 2286/40 (286/12 Mz, disque dur 40 Mo), écran couleur 14 HRCD (piqué 0,29), lecteurs 5"25 et 3"5, nombreux logiciels : 8.000Fr.
Imprimante Citizen 120 D: 6.000 Fr.
Michel BARRAUD. Bur : 42 92 46 36. Dom : 45 47 28

Vends **streamer IRVIN 10 Mo.** Se connecte à la place du lecteur B, 900 Frs. Laisser message au journal, REF 9/9.

Pour les abonnés de DEBUG Magazine, les petites annonces sont gratuites (joindre la bande témoin du dernier numéro reçu). Pour les autres lecteurs, 50 f, par coupon renvoyé. Règlement par chèque ou carte bancaire à l'ordre de: DEBUG Magazine - 3, rue de l'Atlas - 75019 Paris.

J'ACHETE

J'ECHANGE

JE VENDS

DEBUG MAGAZINE

BULLETIN DE PETITES ANNONCES

☐ J'ÉCHANGE ☐ J'ACHÈTE ☐ JE VENDS

☐ Je suis abonné à DEBUG MAGAZINE : ci-joint bande témoin du dernier numéro

☐ Je ne suis pas abonné à DEBUG MAGAZINE; ci-joint un chèque de 50

DEBUG Magazine

MARCHE PC

1/4 page

1 980 F. HT

Dégressifs - 5 Parutions par an : 5 %
10 Parutions par an : 10 %
15 Parutions par an : 15 %
20 Parutions par an : 20 %

1/16 page

490 F. HT

Marché PC de DEBUG Magazine, l'emplacement de choix pour trouver les produits référencés.

1/8 page

990 F. HT

Couleurs de support + 20 %
Frais techniques en sus : forfait de 500 F

TARIFS 1991

1 page : 7 900 F. HT

1/2 page : 3 950 F. HT

LE PROCHAIN NUMERO

Debug Magazine n° 11 *(mars 1991)*

Parution 20 mars 1991

Sujet : *Decouverte d'autres langages*

Dossier : Un interview exclusif avec Bill Gates, président de Microsoft Corp.

Cahier Dr Dobb's Journal :
- KERMIT pour OS/2 (2me partie)
- ADA pour programmeurs PASCAL
- Introduction à MODULA 2

Et Aussi :
- Introduction à C pour programmeurs PASCAL
- Podium
- Idée

Page de John C Dvorak

SONDAGE PERMANENT

Nos lecteurs ont attribué les scores suivants aux articles parus dans DEBUG MAGAZINE n° 9:

1° Un Analyseur de Protocole Série (SPA) par Craig A Lindley (DDJ)

2° Les Fractales par Pierre Morel Fourrier

3° KERMIT rencontre Modula 2 par Brian R Anderson (DDJ)

4° CHAOS - le savoir retrouvé par *ART*

Nous félicitons les auteurs!

ENQUETE

MEILLEURS PRODUITS DE L'ANNEE 1990

Suite à un sondage annoncé dans le numéro 8 (décembre 1990), nos abonnés ont élu les meilleurs produits et services qui, selon eux, ont marqué l'année 1990. Voici les résultats :

MATERIELS

Ordinateur de table : AMSTRAD 1386
Ordinateur portable : AMSTRAD APT386SX
Imprimante matricielle : AMSTRAD LQ 5000 di
Modem : LCE Tel 2
Moniteur : AMSTRAD VGA

LOGICIELS

Système opérationnel : DR DOS 5
Réseau local : Concurrent DOS
Outil de mise au point : SID
Traitement de texte : Evolution Sunset V
Base de Données : Superbase PRO
Tableur : Calcomat
Dessin Assisté : GRA Draft
PAO : GEM Desktop Publisher
Utilitaire : PCTOOLS
Programme de télécom : LCE Com 3

DIVERS

Meilleure documentation : DR DOS 5 Digital Research
Meilleure assistance technique : Coté Ouest
Meilleur magazine informatique : INFO PC - DEBUG - AMSTRAD PC

LA PERSONNALITE DE L'ANNEE : Alain Sugar (Amstrad)

DEBUG MAGAZINE vous invite à faire partager votre expérience avec d'autres lecteurs. Nous sommes intéressés par la publication de textes choisis consacrés à la programmation ou à la technologie en général. Nous pouvons publier vos textes qui apportent une information pertinente à d'autres lecteurs. Votre travail sera rémunéré - demandez notre Dossier Collaborateur.

DEBUG MAGAZINE s'intéresse à vous. Nous aimerions recevoir des informations sur les nouvelles technologies, les nouveaux produits ou les développements de votre entreprise. Envoyez-nous vos communiqués de presse, par courrier ou par télécopie. Et si vous voulez donner un coup de projecteur sur une info (ou une vacherie fondée à dire), ne vous gênez pas.

COMPUMART

Les prix discount !

21, rue de Fécamp - 75012 Paris

LANGUE FORMAT PRODUIT	Prix HT	TTC
ACCESS SOFTWARE		
US 5 HEAVY METAL	239.43	284
US + WORLD CLASS GOLF SIM	274.95	326
ACCOLADE		
US 5 MEAN 18 GOLF	227.54	270
US 5 PINBALL WIZARD	211.50	251
US 5 SERVE AND VOLLEY	213.32	253
ACTION SOFT		
US 5 PERISCOPE	211.50	251
BLUE LION SOFTWARE		
US 5 TICKET TO PARIS	274.95	326
BRODERBUND		
US 5 ANCIENT ART OF WAR	227.54	270
US 5 SHUFFLEPUCK CAFE	211.50	251
US 5 WHERE IN EUROPE IS C	328.19	389
US 5 LODE RUNNER	237.02	281
DOLPHIN MARINE SYS		
US 5 DOLPHIN SAILING SIMU	284.43	337
ELECTRONIC ARTS		
US 5 SCRABBLE	274.95	326
US 5 STARFLIGHT	350.79	416
US 5 CHUCK JAEGER FLIGHT	274.95	326
US 5 GRANDSLAM BRIDGE	310.54	368
EPYX		
US 5 CALIFORNIA GAMES	201.51	239
US 5 SUB BATTLE SIMULATOR	274.95	326
US 5 WINTER GAMES	331.83	394
INFOCOM		
US 5 HITCHHIKER GUIDE GAL	227.54	270
US + BEYOND ZORK	78.18	93
US 5 ZORK TRILOGY	464.56	551
US 5 LEATHER GODDESSES	80.59	96
MICROCOMPUTEUR GAMES		
US 5 RAM	182.33	216
MICROPROSE		
US 5 GUNSHIP	296.31	351
US 5 CRUSADE IN EUROPE	303.39	360
US 5 F15 STRIKE EAGLE	213.32	253
US 5 SILENT SERVICE	213.32	253
US 5 DECISION IN DESERT	274.95	326
US 5 SOLO FLIGHT	274.95	326
US 5 CONFLICT IN VIETNAM	274.95	326
MINDSCAPE		
US 5 BALANCE OF POWER	251.24	298
NEW WORLD COMPUTER		
US 5 MIGHT & MAGIC II	331.83	394
ORIGIN SYSTEMS		
US 5 ULTIMA V WARRIORS DE	350.79	416
PARAGON SOFTWARE		
US 5 DR DOOM'S REVENGE	255.98	304
SIERRA		
US 5 KING QUEST	211.50	251
US 5 HELICOPTER SIMULATOR	211.50	US
5 KING QUEST IV	251.24	298
us 5 SPACE QUEST	211.50	251
US 5 LEISURE SUIT LARRY	201.51	239
us 5 SPACE QUEST 1 SARIEN	255.25	303
US 5 SILPHEED	184.88	219
SOFTWARE TOOLWORKS		
US 5 CHESSMASTER 2100	303.39	360
US + LIFE AND DEATH	303.39	360
SPECTRUM HOLOBYTE		
US 5 SOLITAIRE ROYALE	163.58	194
US 5 SOKO BAN	135.14	251
US 5 GATO	369.76	439
SPINNAKER		
US 5 SARGON III	251.24	298
SUBLOGIC		
US 5 JET SCENERY 3	109.39	130
US 5 JET SIMULATOR 2.1	251.24	298
US 5 JET SCENERY 11	111.44	132
US 5 JET SCENERY 7	109.39	130
us 5 JET SCENERY 1	109.39	130
us 5 JET SCENERY 2	109.39	130
TAITO		
US 5 SKY SHARK	237.02	281
THREE SIX SILICON		
US 5 DARK CASTLE	274.95	326

CONDITIONS DE VENTE :

1. Prix nets, frais de livraison inclus
2. Livraison dans la limite de stocks en 48 heures
3. Paiement seulement par chèque, carte de crédit (Visa, CB. MC) ou mandat
4. A cause de lois sur le copyright, pas de reprise de logiciels ouverts

Je veux recevoir la liste complète des prix.

Nom _____

Société _____

Adresse _____

Ville _____

Code Postal |_|_|_|_|_| Tél. : _____

Backup PRO

rapide, facile et fiable

THE SOFTWARE TOOLWORKS

- 10 Mo en 4 minutes
- jusqu'à 90% de compression de données
- nouvelle interface graphique en couleurs
- mode menu et commande
- vérification et/ou correction automatique d'erreurs
- compatible avec toute unité du DOS et Réseaux Locaux

Le meilleur logiciel de sauvegarde du monde!

OFFRE SPÉCIALE EN CADEAU A CHAQUE ABONNÉ !